UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del título de: INGENIERO DE SISTEMAS

TEMA:

ANÁLISIS, DISEÑO Y DESARROLLO DE UN SISTEMA WEB PARA EL REGISTRO Y GESTIÓN, DE CLIENTES Y ORDENES DE TRABAJO, INTEGRANDO UN MÓDULO DE INFORMACIÓN PARA DISPOSITIVOS CON SISTEMA OPERATIVO ANDROID, PARA LA MECÁNICA ROMERO HNOS. LABOTECA.

AUTORES:

JARA GUTIÉRREZ CRISTIAN PAÚL
RIVERA RIVERA WASHINGTON JAVIER

DIRECTOR:

DÍAZ ORTIZ DANIEL GIOVANNY

Quito, julio del 2014

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además, declaramos que los conceptos y análisis desarrollados, al igual que las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Quito, julio del 2014

_____	_____
Cristian Paul Jara Gutiérrez	Washington Javier Rivera Rivera
CI. 1717153322	CI. 1719325498

DEDICATORIA

El trabajo de titulación dedico a mis padres por su apoyo, consejos, comprensión, amor en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar, y por la formación en valores, principios, carácter, empeño, perseverancia, y mi coraje para conseguir mis objetivos.

Cristian Paúl Jara Gutiérrez

Dedico el presente trabajo primeramente a Dios por haberme permitido llegar hasta ese momento tan importante en mi formación profesional. A mis padres y hermanos que siempre me brindaron su apoyo incondicional.

Washington Javier Rivera Rivera

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Considerando la necesidad de las empresas de ofrecer un servicio de calidad a sus clientes, que brinde una atención cómoda y eficiente, el presente trabajo tiene como objetivo el diseño y desarrollo de una aplicación web para el control de las órdenes de trabajo de la Mecánica Romero Hnos Laboteca, complementando con un módulo de información para el cliente, que interactúa con un aplicativo móvil en Android. Para la comunicación entre la aplicación web y el dispositivo móvil se emplea tecnologías actuales como RESTful para la creación de servicios web y GCM (Google Cloud Messaging) un servicio gratuito de Google para el envío y recepción de mensajes Push.

Para el desarrollo de este proyecto se emplea la metodología ágil Scrum, una metodología que ejecuta el proyecto en bloques cortos y fijos, en donde al final de cada iteración se ofrece al cliente un incremento funcional de la aplicación.

ABSTRACT

Considering the need for companies to provide quality service to its customers, to provide a convenient and efficient service, this thesis aims at designing and developing a web application for controlling work orders of mechanical Romero Hnos. Laboteca, complemented by an information module for the client, which interacts with a mobile application in Android. The project was developed with Spring, a robust and modular framework for creating Java applications. For communication between the web application and Android app, use current technologies such as RESTful, for creating web services and GCM (Google Cloud Messaging) a free Google service for sending and receiving messages Push.

For the development of this project Scrum agile methodology, a methodology that is running the project in short and fixed blocks, where at the end of each iteration the customer provides an increase in functional application is employed.

INTRODUCCIÓN

Antecedentes

La tecnología tiene un papel fundamental dentro de una empresa, puesto que a través de ésta se pueden diseñar y desarrollar herramientas y aplicaciones para administrar los procesos de un negocio, de manera que se optimice tiempo y recursos mejorando la competitividad dentro del mercado y obteniendo mayor retorno de la inversión.

La Mecánica Romero Hnos. Laboteca es una empresa dedicada al mantenimiento y reparación de vehículos livianos, está ubicada en la Av. Maldonado 8618 (entrada a la revisión vehicular) sector Guajaló, desde mayo del 2005 se encuentra atendiendo al público, todos estos años en el negocio le han permitido darse a conocer dentro de su ámbito de trabajo, aumentando considerablemente la demanda de sus servicios; atendiendo alrededor de 10 a 15 vehículos diarios. En la actualidad se ha limitado a registrar la información de sus operaciones en documentos de Microsoft Office como Word o Excel, que si bien son útiles, no ofrecen un óptimo desempeño a gran escala. Con el incremento de clientes, se han presentado varios inconvenientes al momento de utilizar la información referente a los diferentes procesos que realiza la empresa, no contar con una BDD, o algún medio con el que puedan obtener dicha información de forma inmediata, ha generado dificultades en la programación de actividades del personal, en el tiempo de entrega de vehículos y en la comunicación con el cliente. Para compensar esta situación, la empresa adquirió un sistema informático para trabajar sobre aspectos administrativos exclusivamente, descuidando procesos de giro del negocio. Esta área tan importante dentro del contexto empresarial que brinda la posibilidad de gestionar los datos de la empresa como recursos estratégicos de alcance corporativo y a partir de los cuales se podrá generar la información empresarial que las diferentes unidades de la empresa necesitan para operar con efectividad. Por esta razón, se ha visto la necesidad de cubrir esta área con un sistema informático que trabaje sobre el negocio, enfocándose en gestionar las órdenes de trabajo y las tareas dentro de la atención tanto de vehículos como clientes. Otro punto importante que se considerado, es mejorar la atención al cliente, ofreciendo servicios que le permitan relacionarse de manera más cómoda y oportuna. Tomando en cuenta

que la Internet es una herramienta de gran difusión, por medio de la cual las empresas pueden dar a conocer sus productos y servicios a una audiencia más amplia, una aplicación web es la mejor opción para satisfacer las dudas y necesidades de los clientes.

Objetivos

1) Objetivo general

Analizar, diseñar y desarrollar un sistema web para el registro y gestión, de clientes y órdenes de trabajo, integrando un módulo de información para dispositivos con sistema operativo Android, para la mecánica Romero Hnos. Laboteca.

2) Objetivos específicos

I. Evaluar, priorizar y realizar el levantamiento de requerimientos de información concerniente a las necesidades de la mecánica Romero Hnos. Laboteca.

II. Diseñar la arquitectura más conveniente para desarrollar el sistema web y el módulo de información para dispositivos móviles.

III. Diseñar los diagramas y esquemas correspondientes de la metodología Scrum en base a los requerimientos del usuario.

IV. Desarrollar un sistema web para el registro, y gestión de clientes en base a los diagramas y esquemas generados.

V. Diseñar y desarrollar un aplicativo que permita el acceso de información referente al estado de los vehículos a través de dispositivos móviles Android.

VI. Diseñar la integración entre el sistema web y el aplicativo móvil de manera que se pueda obtener el mayor beneficio para los clientes y la mecánica Romero Hnos. Laboteca.

VII. Calendarizar el envío de notificaciones a los clientes mediante la utilización de aplicaciones web y dispositivos móviles.

VIII. Permitir el envío de solicitudes para asignación de citas desde el sistema web y el dispositivo móvil.

IX. Realizar pruebas de funcionamiento del sistema web y el aplicativo móvil que garanticen el correcto manejo de la información tanto en el registro como en la gestión de clientes y órdenes de trabajo.

Justificación

La mecánica Romero Hnos. Laboteca, a lo largo de su operación ha venido manejando la información sobre el historial de los vehículos atendidos, clientes y órdenes de trabajo, de una forma ineficiente, debido a la falta de un sistema informático, lo que le ha conllevado a una serie de problemas al momento de disponer de dicha información. Dados estos antecedentes resulta muy necesaria la elaboración y levantamiento de un sistema web, aplicando los conocimientos adquiridos a lo largo de la carrera de Ingeniería en Sistemas, experiencia laboral e investigación realizada; con ello se pretende lograr la automatización de los procesos de registro y gestión de vehículos y clientes, integrando un módulo de información para dispositivos móviles y que será de gran utilidad para la mecánica y sus clientes.

Con el sistema web se garantizará que la información de clientes, vehículos y trabajos realizados se mantenga actualizada y ordenada, de esta manera la elaboración de cronogramas de atención al cliente puede ser más eficiente.

La satisfacción del cliente es una prioridad, ya que de ello depende el éxito de cualquier negocio, por este motivo se ha considerado agregar un módulo de notificaciones, donde los clientes pueden conocer la información sobre su vehículo de forma exclusiva e inmediata, recibirla vía correo electrónico a su cuenta personal o directamente en su teléfono móvil.

Otra funcionalidad con la que contará el sistema es el envío de ofertas y promociones, hecho que mejoraría la demanda de los servicios de la mecánica. Finalmente, se plantea que la automatización de estos procesos tan importantes, aumentará la productividad de los empleados y brindará una mejor organización de recursos que permitan satisfacer las necesidades tanto de la mecánica como de sus clientes, permitiéndole ofrecer servicios de calidad y mejorar la competitividad en el mercado de la mecánica Romero Hnos. Laboteca.

CAPÍTULO 1
MARCO TEÓRICO

1.1 Conceptos generales

1.1.1 Modelo MVC.

La complejidad de los sistemas crece conforme las necesidades del negocio aumentan, Por esta razón se hace necesario que el equipo encargado del desarrollo sea conformado por varias personas, que en muchos de los casos no tienen mucha experiencia en el área de aplicaciones web, generando varios problemas y necesidades tanto para el equipo desarrollador, como para el administrador del proyecto y el ente que financia el desarrollo del producto.

El patrón MVC (Modelo Vista Controlador) es una arquitectura de diseño software para separar los componentes de aplicación en tres niveles, interfaz de usuario, lógica de control y lógica de negocio. MVC sugiere un enfoque para la creación de sistemas web, basados en la reutilización de código, flexibilidad en la gestión con base de datos y ordenamiento de la estructura de los procesos. (Fuentes & Guevara, 2010)

Un gran número de frameworks de desarrollo para aplicaciones web, como JSF (Java Server Faces), Struts, Javascript entre otros, se encuentran basados en MVC, siendo uno de los más recomendables para el diseño de la aplicación web.

1.1.1.1 Partes de MVC.

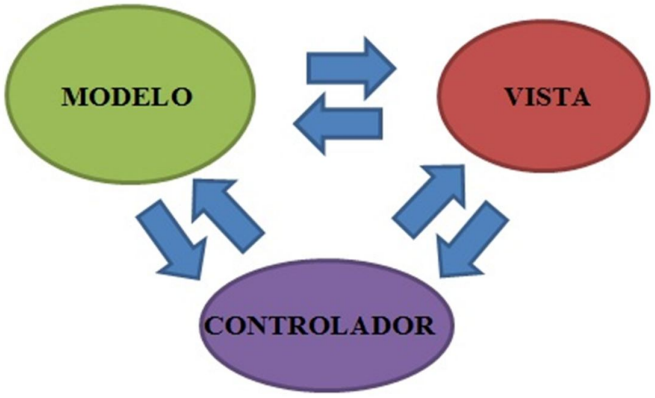
Las capas que plantea el patrón MVC son las siguientes:

Modelo: es el que contiene la lógica de negocio de la aplicación y la que toma decisiones sobre el estado de los objetos dentro del sistema. Se encarga de ejecutar los cambios en la aplicación. Accede a la capa de persistencia pero mantiene su independencia mediante un débil acoplamiento, también se encarga de notificar a la vista de los cambios que se realizan sobre la capa de persistencia.

Vista: es la encargada de mostrar información al usuario y recibe su interacción, también pueden dar el servicio de actualización para que interactúe con el controlador o con el modelo.

Controlador: es el que recibe e interpreta la interacción del usuario, actuando sobre el modelo y vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización. (Gómez, 2011)

Figura 1. Flujo del Modelo Vista Controlador



Elaborado por: Paul Jara & Javier Rivera.

1.1.2 Aplicación web.

En ingeniería de software se denomina aplicación web a los sistemas informáticos que los usuarios pueden utilizar a través de Internet o de una intranet mediante el uso de un navegador, su uso se ha popularizado debido a la expansión del Internet y debido a lo práctico del navegador web como cliente ligero. (Ávila, 2009)

- **Ventajas**

- Ahorra tiempo, se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.

- No hay problemas de compatibilidad, basta tener un navegador actualizado para poder utilizarlas.

- No ocupan espacio en el disco duro.

- Actualizaciones inmediatas, como el software lo gestiona el propio desarrollador, cuando se conecta usa siempre la última versión publicada.

- Bajo consumo de recursos, dado que toda o gran parte de la aplicación no se encuentra sobre el ordenador cliente, este no consume mayormente recursos. Las tareas pesadas las realiza el servidor web.

- Multiplataforma, se puede usar desde cualquier sistema operativo que disponga de un navegador web.

- Portables: es independiente del ordenador donde se utilice (un PC de sobremesa, un portátil...) porque se accede a través de una página web (solo es necesario disponer de acceso a Internet). La tendencia de acceso a las aplicaciones web a través de teléfonos móviles requiere un diseño específico de los ficheros CSS para facilitar el acceso desde estos dispositivos.

- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones web enriquecidas (RIA).

➢ **Desventajas**

- Limitación de funcionalidades: por lo general las aplicaciones web prestan menos funcionalidades que una aplicación de escritorio, aunque la brecha es cada vez más pequeña gracias a las aplicaciones web enriquecidas.

- Dependencia de terceros: la tendencia a depender de un tercero para la administración de los servicios y/o la información, hace que la aplicación sea dependiente de las políticas de la empresa proveedora.

1.1.3 Aplicaciones móviles.

Son aplicaciones que se pueden descargar o instalar directamente en el dispositivo móvil, realizan alguna o varias tareas específicas y cuentan con varios repositorios o tiendas en Internet que permiten descargarlas de manera pagada o gratuita.

En la actualidad existen varios sistemas operativos para dispositivos móviles, por ejemplo Android, IOS, Windows Phone entre otros, el alcance de este trabajo se enfocará únicamente en las aplicaciones que se ejecuten sobre Android.

1.1.4 Cloud Computing.

> El Cloud Computing, también conocido como "Nube", ha sido definido por el NIST (National Institute of Standards and Technology) como un modelo de servicios escalables bajo demanda para la asignación y el consumo de recursos de cómputo. Describe el uso de infraestructura, aplicaciones, información y una serie de servicios compuestos por reservas de recursos de computación, redes, información y almacenamiento. (Río, 2012)

Cloud Computing ofrece servicios en la Nube, montando las aplicaciones en hardware de terceros, permite al proveedor de contenidos prescindir de la instalación y mantenimiento de cualquier tipo de hardware, además del coste de licencias y el uso fraudulento de software que el proveedor de la infraestructura brinda. Con lo que el despliegue inicial de las aplicaciones en la Nube tendrá un coste menor, mayor disponibilidad y contribuirán con la ecología ya que existe ahorro de hardware, software, energía y de personal para el mantenimiento de la infraestructura. Una vez que se popularizó el uso de las computadoras personales el uso de este tipo de tecnologías quedo prácticamente en desuso, es por eso que ciertas personas califican este tipo de sistemas como una tendencia. En la actualidad conforme el uso de dispositivos móviles con conexión a Internet va en aumento, también aumenta el uso de Cloud Computing, además muchas empresas están migrando sus sistemas y datos a la Nube.

1.1.4.1 Aplicaciones de Cloud Computing.

Esta tecnología permite acceder a servicios y aplicaciones a través de Internet, para lo cual utilizan un navegador, de esta manera el usuario podría hacer uso del servicio sin necesidad de instalar ningún tipo de software en el dispositivo. Por ejemplo Google Docs, permite crear documentos, hojas de cálculo y presentaciones sin necesidad de instalar algún software de ofimática, otros ejemplos son los antivirus online, las emisoras de radio y video, etc. Con el ejemplo de Google Doc se obtiene la ventaja de que los documentos pueden quedar grabados fuera del dispositivo y se los pueda abrir desde cualquier parte, simplemente se necesita conexión a Internet. La tendencia a usar este tipo de servicios cada vez es mayor y los dispositivos móviles son los mayores beneficiados ya que, al no contar con hardware suficiente para ejecutar este tipo de aplicaciones encuentran una alternativa perfecta en los servicios online o Cloud Computing

1.1.4.2 Modelos de servicios de Cloud Computing.

- **Software as a Service (SaaS):** en español Software como Servicio. Modelo de distribución de software donde una empresa sirve el mantenimiento, soporte y operación que usará el cliente durante el tiempo que haya contratado el servicio. El cliente usará el sistema alojado por esa empresa, la cual mantendrá la información del cliente en sus sistemas y proveerá los recursos necesarios para explotar esa información. Ejemplos: Salesforce, Basecamp.

- **Infrastructure as a Service (IaaS):** en español Infraestructura como Servicio. Modelo de distribución de infraestructura de computación como un servicio, normalmente mediante una plataforma de virtualización. En vez de adquirir servidores, espacio en un centro de datos o equipamiento de redes, los clientes compran todos estos recursos a un proveedor de servicios externo. Una diferencia fundamental con el hosting virtual es que el aprovisionamiento de estos servicios se hacen de manera integral a través de la web. Ejemplos: Amazon Web Services EC2 y GoGrid.

- **Platform as a Service (PaaS):** en español Plataforma como Servicio. Aunque suele identificarse como una evolución de SaaS, es más bien un modelo en el que se ofrece todo lo necesario para soportar el ciclo de vida completo de construcción y puesta en marcha de aplicaciones y servicios web, completamente disponibles en la Internet. Otra característica importante es que no hay que instalar el software en los equipos de los usuarios. PaaS ofrece múltiples servicios, pero todos aprovisionados como una solución integral en la web. (Otero, 2013)

1.1.5 JBoss 7.

JBOSS 7 es la última versión de este servidor de aplicaciones a la fecha de la redacción de este documento, apareció el 15 de febrero del 2012, se proyecta como una herramienta de rápida y poderosa implementación de las especificaciones de JEE 6.

1.1.5.1 Estructura de directorios.

La estructura de ficheros en JBoss 7 muestra la forma en la que se encuentran distribuidos y organizados los componentes del servidor de aplicaciones.

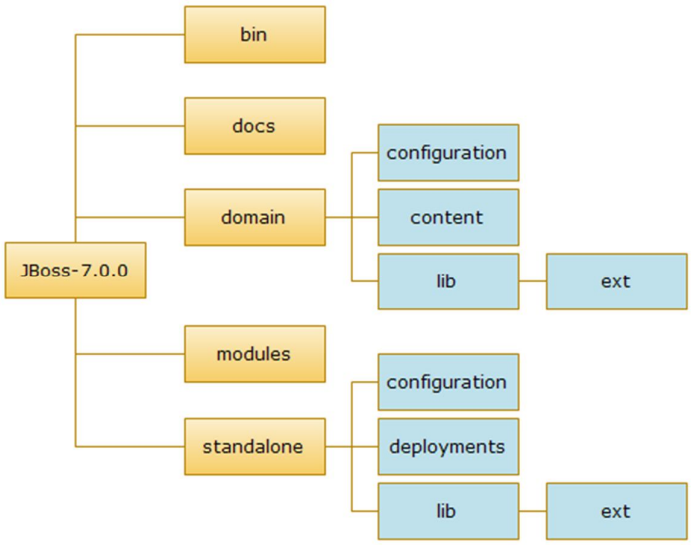
Tabla 1. Estructura de ficheros en JBoss 7

Directorio	Descripción
Bin	Scripts de arranque con sus archivos de configuración.
Bundles	OSGi bundles
docs/schema	Esquemas XML y archivos de definición
Domain	Archivos de configuración, contenido desplegado, y áreas escribibles por procesos del modo dominio.
Modules	Como JBoss AS 7 está basado en una arquitectura de carga de clases (classloading) modular, varios módulos usados en el servidor se encuentran aquí.
Standalone	Archivos de configuración, contenido desplegado, y áreas escribibles por server en modo independiente (Standalone).
welcome-content	Página de bienvenida por defecto

Fuente: docs.jboss.org

En la figura 2 se puede identificar los nodos del servidor y su estructura interna, así como la subdivisión de los nodos principales Standalone y Domain. (Marchioni, Mastertheboss, 2012)

Figura 2. División de la estructura de ficheros en JBoss 7



Fuente: (Marchioni, JBoss AS 7, 2011, p. 19)

1.1.5.2 Administración.

Para el control de su entorno JBoss 7 presenta varias opciones detalladas a continuación:

Consola de administración web.

- Un cliente muy completo de línea de comando (también llamado solo CLI).
- Una API Java que se puede acceder directamente con Java Remoting.
- Una API REST para enviar comandos por HTTP.

1.1.5.3 Modos de operación.

En JBoss AS 7 existen dos formas de operación:

- **Standalone (independiente)** para muchos casos, la capacidad de manejo centralizado no es necesaria por esto, una instancia de JBoss 7 se puede ejecutar como un Standalone.

  Una instancia de Standalone es un proceso independiente, se ejecuta usando el script de inicio standalone.sh (standalone.bat para Windows). Para ejecutar más de una instancia en modo Standalone la responsabilidad de la administración de los nodos queda del lado del administrador.

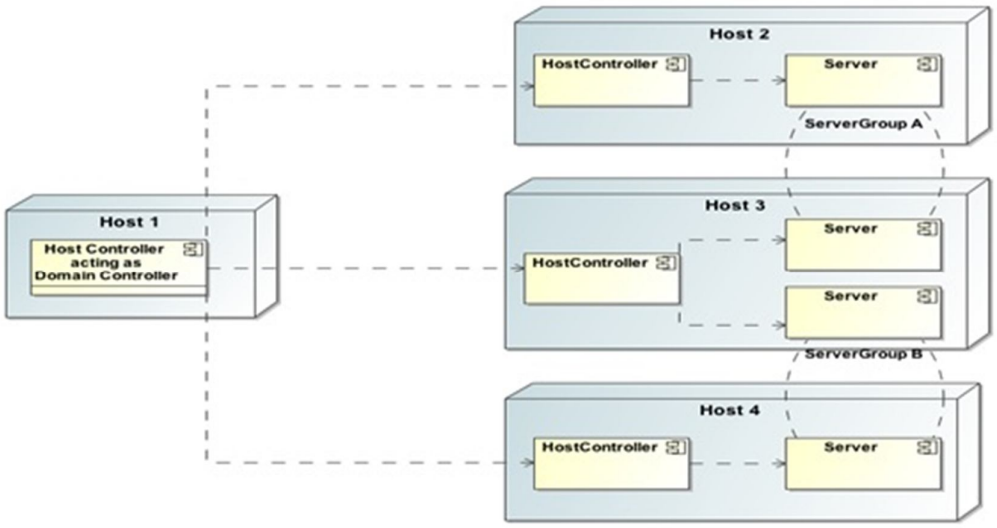
- **Domain (dominio)** el concepto de dominio puede ser en un principio un poco difícil de entender. La razón de esto es que en el paradigma Java EE, se utiliza para tratar con los servidores en lugar de los dominios. Esto es más común con los desarrolladores.

  Básicamente, un dominio es una unidad administrativa. Es un perímetro dentro del cual todos los servidores JBoss 7 son administrados por un controlador de dominio.

  Es importante entender que el concepto de dominio no interfiere con las capacidades entregadas por los administradores de servidores. Por ejemplo, es posible configurar un dominio de nodos de servidor (Domain) que se ejecutan en un clúster, proporcionar balanceo de carga y alta disponibilidad. Pero, puede ser que también prestan los mismos servicios con un conjunto de servidores de aplicaciones independientes (Standalone).

  Lo que diferencia en estos dos escenarios es que cuando se ejecuta en un dominio, se los puede administrar en conjunto de manera eficiente desde una unidad centralizada. Por otro lado, la gestión de un conjunto de instancias independientes a menudo requiere capacidades de gestión de varios servidores sofisticados, que no siempre están al alcance de las empresas. (Marchioni, JBoss AS 7, 2011)

Figura 3. Topología Domain



Fuente: docs.jboss.org

1.2 Java

Java es un lenguaje de programación orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas presentaciones, multitarea y dinámico. Es la tecnología subyacente que permite el uso de programas punteros, como herramientas, juegos y aplicaciones de negocios. Java se ejecuta en más de ochocientos cincuenta millones de ordenadores personales de todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y aparatos de televisión. (Java, 2013)

- **Ventajas**

- La compatibilidad: un programa Java es indiferente a la arquitectura sobre la que se ejecuta por lo cual no es necesario reescribir código para cada navegador o dispositivo, la mayor parte de navegadores actualmente tienen compatibilidad con Java.

- Es un lenguaje de programación orientado a objetos con características como herencia, abstracción, polimorfismo y encapsulamiento, esenciales para un

desarrollo ágil y ordenado que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.

- Al ser Java un lenguaje de programación, se puede realizar cálculos matemáticos, procesadores de palabras, interactuar con bases de datos, aplicaciones gráficas, animaciones, sonido, hojas de cálculo, etc., por otra parte las páginas web realizadas en este lenguaje tienden a no ser estáticas, ya que se dispone de la facilidad de incorporar elementos multimedia y les otorga un nivel alto de interactividad.

- El JDK es una herramienta libre, creada por la empresa Sun Microsystem, actualmente propiedad de Oracle Corporation, respaldada por un gran número de proveedores como RedHat, IBM, Motorola, Samsung entre otros.

- Actualmente se dispone de herramientas como iText y Jasper Reports que facilitan en gran medida la creación de reportes tanto para ambiente de escritorio como web.

➢ **Desventajas**

- El ser un lenguaje de programación compilado requiere de un entorno de ejecución para sus programas, conocido como JRE (Java Runtime Environment).

- Lentitud frente a un programa equivalente escrito en C o C++ ya que su entorno de ejecución (JRE) le agrega tiempo de procesamiento.

- A pesar de que Java ofrece herramientas como AWT para la generación de imágenes, estas son muy costosas en términos de procesamiento y terminan siendo muy lentas.

1.2.1 Uso de la tecnología Java.

Hasta la fecha, la plataforma Java ha atraído a más de 9 millones de desarrolladores en todo el mundo, con características como versatilidad, eficiencia, portabilidad de su plataforma y seguridad las cuales la hacen idónea para utilizarla en todos los sectores de la industria en el mundo. (Oracle, Java, 2011)

"Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes". (Oracle, Java, 2011)

- Las cifras de uso de Java en dispositivos es sumamente alta.
- El 97% de los escritorios empresariales y el 89% de los escritorios (o computadoras) en Estados Unidos, 3 mil millones de teléfonos móviles ejecutan Java,
- El 100% de los reproductores de Blue-ray incluyen Java
- 5 mil millones de Java Cards en uso.
- 125 millones de dispositivos de televisión ejecutan Java.
- 5 de los 5 principales fabricantes de equipos originales utilizan Java ME.

(Oracle, Java, 2011)

1.3 Java EE

Es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar aplicaciones empresariales basadas en web. Java EE incluye muchos componentes de Java Standard Edition (Java SE). La plataforma Java EE consta de un conjunto de servicios, API y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones basadas en web de varios niveles.

Java EE simplifica el desarrollo de aplicaciones y reduce la necesidad de programación y formación para programadores al crear componentes modulares normalizados y reutilizables, así como al permitir controlar muchos aspectos de la programación automáticamente por nivel.

Si es un desarrollador empresarial, necesita Java EE. Los desarrolladores empresariales necesitan Java EE porque crear aplicaciones empresariales distribuidas no es sencillo, y necesitan una solución de alta productividad que les permita centrarse únicamente en escribir su lógica empresarial y disponer de una gama completa de servicios de clase empresarial en la que confiar, como objetos distribuidos transaccionales, middleware orientado a mensajes y servicios de directorio y asignación de nombres. (Oracle, Java, 2011)

1.3.1 Definición.

Es el conjunto de especificaciones y prácticas coordinadas, que permiten soluciones para el desarrollo, implantación y administración de aplicaciones de múltiples capas, con un servidor centralizado. Por otra parte añade las capacidades necesarias para proveer una plataforma Java completa, estable, segura y rápida a nivel empresarial. Su característica principal es que reduce significativamente el costo y complejidad en el desarrollo de soluciones de múltiples capas, lo que resulta en servicios que pueden ser rápidamente implementados y fácilmente distribuidos.

1.3.2 Características.

Entre las principales características de Java EE encontramos las siguientes.

- Programación eficiente: desarrollo de forma estándar para construir aplicaciones en N capas (cliente, servidor web, etc.) reutilizando código.

- Extensibilidad: frente a la demanda del negocio permite utilizar varias soluciones para llegar al mismo objetivo.

- Integración: los equipos de ingeniería precisan estándares que favorezcan la integración entre diversas capas de software.

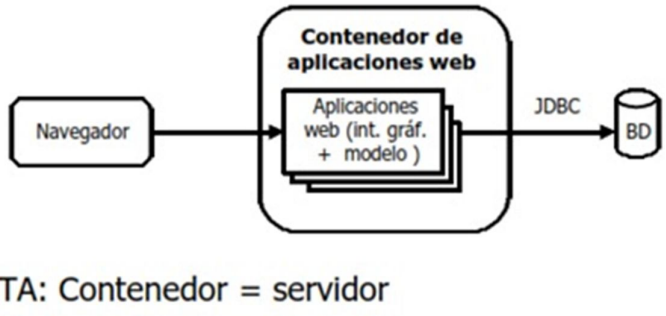
- Basado en estándares: el lenguaje Java y la tecnología relacionada evolucionan a través de Java Community Process, un mecanismo que permite desarrollar especificaciones técnicas para la tecnología Java. (Oracle, 2012)

- Seguridad: Java EE ofrece un entorno de aplicaciones avanzado con un alto nivel de seguridad que es idóneo para las aplicaciones de red. (Oracle, 2012)

1.3.3 Contenedores Java EE.

Los contenedores proveen un entorno de ejecución para componentes Java, también brindan los servicios de seguridad, transacciones, administración de ciclo de vida, caching, persistencia, comunicación en red.

Figura 4. Principales tecnologías proporcionadas por Java EE



Fuente:tic.udc.es

javax.ejb.*: Se la utiliza para definir un conjunto de APIs que un contenedor de objetos distribuidos soportará para suministrar persistencia, RPCs, control de concurrencia, transacciones y control de acceso para objetos distribuidos.

javax.naming: Define la API de Java Naming and Directory Interfaces (JNDI).

java.sql: Los paquetes Java.sql y Javax.sql definen la API de JDBC.

java.transaction.*: Estos paquetes definen la Java Transaction API (JTA).

java.xml.*: Definen la API JMS.

java.persistence: Provee las clases e interfaces para gestionar la interacción entre los proveedores de persistencia, las clases administradas y los clientes de la Java Persistence API. Gracias a ellos se reducen los tiempos de desarrollo y se simplifica el mantenimiento.

1.4 Spring

Spring cuenta con muchas aplicaciones de las diferentes tecnologías de código abierto, todas las cuales se unifican bajo el nombre de Spring Framework. Cuando se trabaja con Spring, un desarrollador de aplicaciones puede utilizar una gran variedad de herramientas de código abierto, sin necesidad de escribir páginas y páginas de código y sin acoplamiento a ninguna herramienta en particular. (Clarence & Harrop, 2012, p. 1)

Es framework liviano para la construcción de aplicaciones, fue lanzado bajo la versión de apache 2.0 en 2003, no necesita un modelo de programación en particular, ofrece una amplia libertad a los programadores y soluciones fáciles de usar, se lo puede aplicar en cualquier desarrollo Java e integrar con varios framework y herramientas como JSF, Struts, Hibernate, JPA, JDBC, etc.

1.4.1 Definición.

Spring Framework es un framework liviano de código abierto, está basado en la técnica Inversión de Control (IoC) que promueve el bajo acoplamiento a partir de la inyección de dependencias entre los objetos y una implementación de programación orientada a aspectos (AOP), que simplifica el desarrollo de aplicaciones a través de la utilización de módulos, y reduce la complejidad del software empresarial de Java EE.

1.4.2 Características funcionales.

Simplicidad: simplifica el trabajo para los desarrolladores de aplicaciones Java ya que evita el uso de EJBs. “En su estado actual EJB es complicado. Es complicado porque EJB fue creado para resolver cosas complicadas, como objetos distribuidos y transacciones remotas.” (Walls, 2011)

POJO (Plain Old Java Object): desarrollo ligero y mínimamente invasivo con objetos (POJO) que no obligan al desarrollador a crear implementaciones de interfaces que exceden las necesidades de la aplicación o incrementan la complejidad de la misma.

XML: permite la configuración basada en archivos XML de manera que en caso de requerirse un cambio en la configuración, no es necesario recompilar el código Java, sino simplemente modificar el fichero XML.

Seguridad: requisito no funcional, implementado como aspecto AOP a través del framework Acegi.

Remoto: cuenta con una implementación de RMI que se encuentra simplificado para un uso más sencillo, además acceso y publicación para RESTful.

Pruebas: provee un paquete para pruebas (testing), específico para componentes del framework que se encuentra integrado con JUnit.

1.4.3 Arquitectura.

DAO: permite el acceso JDBC con manejo de transacciones (desde el módulo AOP).

ORM: permite la interacción con servicios empresariales.

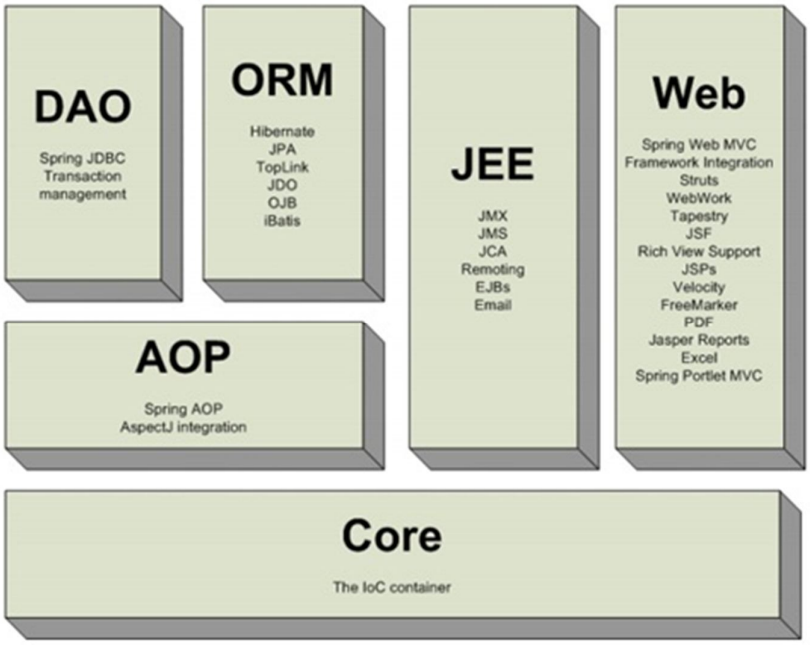
Java EE: acceso e interacción con servicios empresariales.

Web: provee un contexto apropiado para el desarrollo web e integración con otros frameworks como Struts, JSF, Tapestry, etc.

AOP: se utiliza para modularizar cuestiones transversales de una aplicación, para ponerlo de una manera simple, es solo un interceptor para algunos procesos, por ejemplo, cuando un método se llama “ejecutar” Spring AOP puede tomar el método “ejecutar” y añadir funcionalidad extra antes o después de la ejecución del método. (Mkyong, 2010)

Core: la fábrica de beans (BeanFactory) utiliza el patrón de inversión de control (Inversion of Control) y configura los objetos a través de la inyección de dependencia (Dependency Inyection) (Ceresola Millet , 2012, p. 28).

Figura 5. Arquitectura Spring



Fuente: palermo.edu

1.5 Hibernate

Hibernate es una herramienta que facilita la unión entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las BDD en entornos Java, el término utilizado para esto es ORM (Object-Relational mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

Permite a la aplicación manipular los datos en la BDD operando sobre objetos, con todas las características de la POO, además genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre los motores de BDD más utilizados con un ligero incremento en el tiempo de ejecución, de esta manera reduciendo el tiempo de desarrollo.

1.5.1 Características.

- Está diseñado para ser flexible y poder adaptar su uso sobre una BDD ya existente.

- Tiene la funcionalidad de crear las entidades en la BDD a partir de la información disponible.

- Ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "Criteria Query").

- Puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA y es parte estándar de esta plataforma.

1.5.2 Arquitectura

En la figura 6 se muestran los roles de las interfaces de Hibernate más importantes en las capas de persistencia y de negocio de una aplicación Java EE.

El API de Hibernate es una arquitectura de dos capas (Capa de persistencia y Capa de Negocio).

Las Interfaces mostradas se clasifican de la siguiente forma:

- Interfaces llamadas por la aplicación para realizar operaciones básicas:

  - ✓ Session: interfaz primaria utilizada en cualquier aplicación Hibernate (SessionFactory).

  - ✓ Transaction: permite a la aplicación definir las unidades de trabajo, mientras mantiene abstracción de la implementación de la transacción subyacente. (JBoss, 2012)

  - ✓ Query: permite realizar peticiones a la BDD y controla cómo se ejecuta dicha petición (query). Las peticiones se escriben en HQL o en el dialecto SQL nativo de la BDD que se está utilizando. Una instancia de la interface Query se utiliza para enlazar los parámetros de la petición, limitar el número de resultados devueltos por la petición y para ejecutar dicha petición.

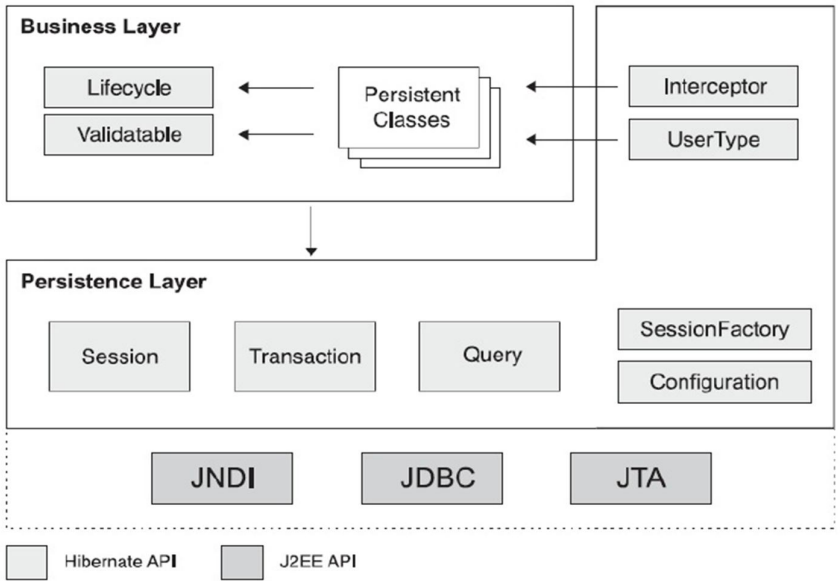
- Interfaces llamadas por el código de la infraestructura de la aplicación para configurar Hibernate.

- La más importante es la clase Configuration: se utiliza para configurar e iniciar Hibernate. La aplicación utiliza una instancia de Configuration para especificar la ubicación de los documentos que indican el mapeado de los objetos y propiedades específicas de Hibernate y a continuación crea la Session Factory.

- Interfaces callback que permiten a la aplicación reaccionar ante determinados eventos que ocurren dentro de la aplicación, tales como Interceptor, Lifecycle y Validatable.

- Interfaces que permiten extender las funcionalidades de mapeado de Hibernate, como por ejemplo UserType, CompositeUserType e IdentifierGenerator.

Figura 6. Arquitectura de Hibernate



Fuente: unife.edu.pe

1.6 Android

Android es un sistema operativo orientado a dispositivos móviles, está basado en Linux, un núcleo de sistema operativo libre, gratuito y multitarea. Estas características lo han convertido en el más popular en la actualidad. Android proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java. (Molina, 2012)

Su gran crecimiento se debe principalmente a la infinidad de aplicaciones disponibles y herramientas de programación gratuitas para la creación de las mismas. Esto atrae

a los fabricantes de dispositivos móviles por el bajo costo que representa su implementación.

- **Ventajas.**

- Disponibilidad de instalarse en una gran variedad de smartphones, siendo una apuesta importante por fabricantes y operadoras, pudiendo adaptarse a todo tipo de necesidades y en una amplia gama de precios.

- Existe una infinidad de aplicaciones que pueden ser descargados desde Google Play, que es la tienda de Google donde aproximadamente dos tercios de los aplicativos son gratuitos, aunque existen otros repositorios como Samsung Apps, Aptoide, etc

- Código abierto, libre de licencias, pudiendo así los programadores perfeccionar cada vez más este sistema operativo.

- No se encuentra limitado solo a ciertas operadoras, Android no tiene fronteras, permitiendo que cualquier operadora haga uso del mismo.

- Android es completamente personalizable con la posibilidad de crear sus propias capas como MotoBlu o HTC Sense, permitiendo a unos y a otros poder personalizar sus teléfonos de la mejor manera posible y dando a elegir al usuario la interfaz más adecuada para su gusto.

- Android es un sistema operativo multitarea, capaz de gestionar varias aplicaciones abiertas a la vez dejando en suspensión aquellas que no se utilicen.

- Actualmente, Android se encuentra funcionando en varios tipos de dispositivos tales como teléfonos celulares, televisores inteligentes, tablets, automóviles, etc.

- **Desventajas.**

- Debido a las características de hardware de la mayoría de teléfonos táctiles, el consumo de batería es mayor.

- La necesidad de instalar aplicaciones externas para solucionar problemas de uso normal.

- Android está totalmente fragmentado provocando problemas de incompatibilidad con algunas aplicaciones de Market que funcionan en determinadas versiones de Android.

1.6.1 Arquitectura de Android.

Como se muestra en la figura 7 Android tiene una arquitectura de varias capas, las que interactúan entre sí, cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.

Aplicaciones: este nivel contiene tanto las aplicaciones por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo.

Framework de aplicaciones: representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Todas las aplicaciones utilizan el mismo conjunto de API y el mismo Framework, representado por este nivel.

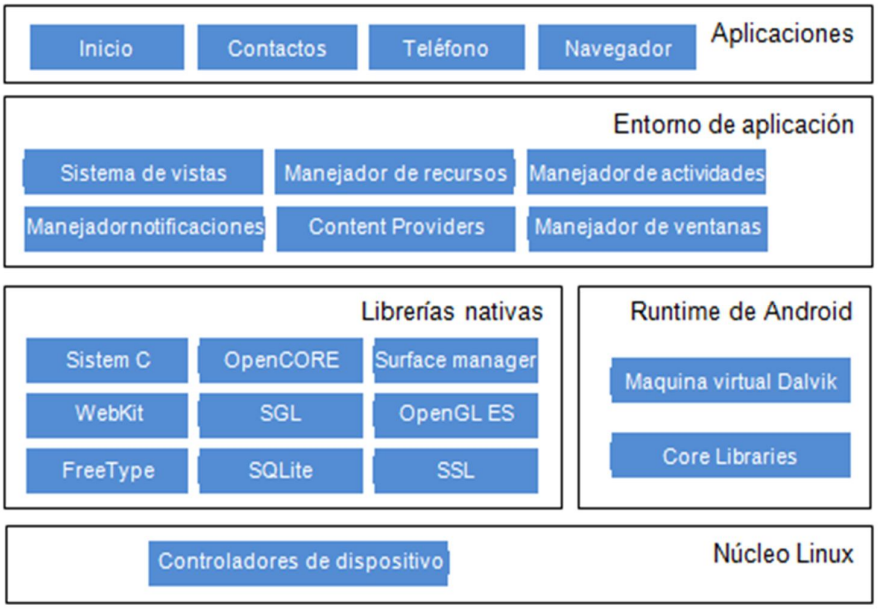
Librerías: incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador, junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Android Runtime: se encuentra al mismo nivel que las librerías de Android. Está basado en el concepto de máquina virtual utilizado en Java. Dado las limitaciones de los dispositivos donde ha de correr Android no fue posible utilizar una máquina

virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones. (androidcurso, 2011)

Núcleo Linux: Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. (androidcurso, 2011)

Figura 7. Arquitectura Android



Fuente: .androidcurso.com

1.6.2 Versiones.

Android ha lanzado varias versiones desde su aparición, a continuación se detallan ordenadas cronológicamente. (androidcurso, 2011)

Tabla 2. Versiones de Android

Nombre	Versión	Nivel de API	Fecha De Lanzamiento
Base	Android 1.0	1	Septiembre 2008
Base 1.1	Android 1.1	2	Febrero 2009
CupCake	Android 1.5	3	Abril 2009
Donut	Android 1.6	4	Septiembre 2009
Éclair	Android 2.0	5	Octubre 2009
	Android 2.1	7	Enero 2010
Froyo	Android 2.2	8	Mayo 2010
Gingerbread	Android 2.3	9	Diciembre 2010
Honeycomb	Android 3.0	11	Febrero 2011
	Android 3.1	12	Mayo 2011
	Android 3.2	13	Julio 2011
Ice Cream Sandwich	Android 4.0	14	Octubre 2011
	Android 4.0.1	15	Diciembre 2011
Jelly Bean	Android 4.1	16	Julio 2012
	Android 4.2	17	Noviembre 2013
	Android 4.3	18	Julio 2013
KitKat	Android 4.4	19	Octubre 2013

Elaborado por: Paul Jara & Javier Rivera.

1.6.3 Aplicativo Android.

Un aplicativo Android es un programa diseñado para permitir al usuario realizar una o varias tareas, utilizando el hardware del dispositivo que use Android como sistema operativo.

Las aplicaciones Android se desarrollan habitualmente en lenguaje Java con Android SDK (Software Development Kit), un kit de desarrollo provisto gratuitamente por Google con una serie de herramientas que son de gran ayuda para el desarrollador. Las aplicaciones se encuentran comprimidas en formato APK, que se pueden instalar sin dificultad desde cualquier explorador de archivos en la mayoría de dispositivos.

Actualmente un punto importante a considerar en la creación de aplicaciones Android es la conexión con servidores web para el intercambio de información, la

forma más usada para este fin es mediante un servicio web. Existen varias arquitecturas que permiten la creación de servicios web, este es el caso de RESTful.

RESTful (Representational State Transfer) define varios principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios por tener un estilo bastante más simple de usar, comparado con SOAP y las interfaces basadas en WSDL. (Rodriguez, 2008)

1.7 Metodología Scrum

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, tiene como objetivo principal elevar al máximo la productividad de un equipo. Está basada en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación. (Manami, 2009)

- **Ventajas.**

- Cumplimento de expectativas: el cliente establece sus expectativas indicando el valor que le aporta cada requisito, el equipo los estima y con esta información el Product Owner establece su prioridad. De manera regular, en las demos de Sprint el Product Owner comprueba que efectivamente los requisitos se han cumplido.

- Flexibilidad a cambios: alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.

- Reducción del Time to Market: el cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.

- Mayor productividad: se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.

- Maximiza el retorno de la inversión (ROI): producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.

- Predicciones de tiempos: mediante esta metodología se conoce la velocidad media del equipo por Sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el Backlog. (Manami, 2009)

1.7.1 Componentes Scrum.

- **Product Backlog:** es una lista inicial de requerimientos de los usuarios y propietarios del sistema. Se considera un documento dinámico que incorpora las constantes necesidades del sistema y se mantiene durante todo el ciclo de vida.

- **Sprint:** es la base para el desarrollo, su duración máxima aconsejable es de 30 días, En este punto se llevan a cabo tareas pre-establecidas y no se puede modificar el trabajo acordado en el Backlog. Solo el ScrumMaster puede abortar un Sprint si lo considera no viable.

- **Spring Backlog:** especifica la serie de tareas que se van a desarrollar según los requisitos señalados, estas tareas tiene una duración de entre 4 y 6 meses.

Al final del Sprint se busca un incremento en la funcionalidad. (Albaladejo, 2012)

1.7.2 Actores.

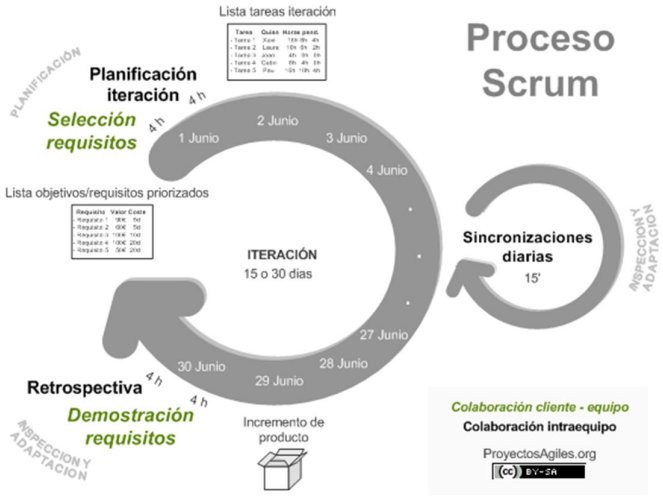
- **Propietario del producto (Product Owner)**: representa a todos los interesados en el producto final, este actor es el responsable de las áreas de financiación, retorno de la inversión y lanzamiento del proyecto.

- **Equipo:** es el responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software, el cual debe tener las siguientes características:

  - ✓ Auto-gestionado.
  - ✓ Auto-organizado.
  - ✓ Multi-funcional.

- **Scrum Master:** es el responsable de la formación y entrenamiento del proceso, incorporación de Scrum en la cultura de la empresa y garantía de cumplimiento de roles y responsabilidad. (Albaladejo, 2012)

Figura 8. Flujo de Scrum



Fuente: proyectosagiles.org

CAPÍTULO 2
DISEÑO DEL SISTEMA WEB

En este capítulo se describe los puntos considerados dentro del análisis y diseño del sistema web, en los cuales recae la importancia del éxito del proyecto, el planteamiento de estos puntos de forma correcta permitirá elaborar el sistema en menor tiempo y con resultados óptimos, de manera que satisfagan las necesidades planteadas por el cliente. Además, ofrecer una mejor visión y comprensión de los procesos que involucra el sistema.

Dentro del análisis se presenta como factor importante el cliente, quien provee de información y criterios, acerca de los requerimientos y resultados esperados dentro de cada uno de los procesos y funcionalidades.

2.1 Recopilación de los requerimientos del sistema

La mecánica Romero Hnos. Laboteca requiere de un sistema que permita una correcta gestión de los vehículos atendidos. Además, de establecer mecanismos para una comunicación eficiente con los clientes. Mediante el proceso de análisis se determinó varias necesidades:

- Diseño de una BDD para el negocio.
- Automatizar el proceso de registro y modificación de clientes.
- Automatizar el proceso de registro y modificación de vehículos.
- Automatizar el proceso de registro y modificación de fichas de ingreso de vehículos a talleres.
- Generar una orden de trabajo por cada ficha de ingreso.
- Registrar un historial de los vehículos atendidos.
- Ofrecer información al cliente de promociones y estados de los vehículos.
- Controlar las actividades realizadas por los mecánicos mediante las órdenes de trabajo.

- Evaluar el tiempo empleado en cada trabajo realizado.

Solución:

Considerando los puntos anteriormente expuestos se requiere la elaboración de un sistema web que incorpore los siguientes módulos:

- **Módulo de seguridad:** la seguridad es una parte fundamental de todo software, por esta razón debe incorporarse un módulo de seguridad, configurable y administrable, que proporcione al administrador el control de acceso a los usuarios y funcionalidades. Además, permita una fácil configuración de los esquemas involucrados.

- **Módulo de configuración:** permitir, al administrador y al implementador del sistema gestionar los parámetros desde una interfaz amigable y rápida.

- **Módulo de gestión de clientes:** permitir de manera automática, obtener datos de los clientes y vehículos que se encuentren registrados en el sistema, además de un fácil y rápido manejo de información como, el historial de reparaciones, mantenimientos o revisiones realizadas a los vehículos, además permitir la recepción de solicitudes de citas.

- **Módulo de operaciones:** permitir, al personal de la mecánica, crear y gestionar las órdenes de trabajo de manera fácil y rápida, para brindar un mejor servicio.

- **Módulo de promociones y notificaciones a clientes**: permitir, al personal de la mecánica, configurar el envío de notificaciones a los clientes de manera programada y automática, sobre información que la empresa considere relevante o que el cliente haya solicitado.

- **Módulo web para clientes:** permitir a los clientes interactuar con el sistema, visualizando información de su vehículo a través de un navegador.

2.2 Arquitectura del sistema web

Para el diseño de la arquitectura del sistema web, inicialmente se determinó las necesidades y restricciones bajo las cuales se va a ejecutar.

2.2.1 Requerimientos.

Tabla 3. Requerimientos de la arquitectura

Atributo de calidad	Requerimiento de arquitectura
Velocidad	Se requiere obtener un tiempo de respuesta de menos de 3 segundos en el 90% de las peticiones.
Seguridad	Todas las comunicaciones deben ser autenticadas.
Persistencia de datos	Los datos ingresados en el sistema deben ser guardados de manera que después se pueda acceder a ellos.
Usabilidad	La interfaz de usuario debe ejecutarse sobre un navegador web, que tenga acceso a la red.
Escalabilidad	La aplicación debe ser capaz de soportar al menos 40 usuarios conectados al mismo tiempo.
Configurable	La aplicación debe ser configurable de manera que ciertos valores usados para su ejecución puedan ser cambiados por el usuario administrador.
Tolerancia a fallos	La aplicación debe ser capaz de mantener la integridad de datos frente a un error que se presente durante su ejecución.
Reusabilidad	La aplicación debe ser escrita de manera modular de tal forma que si se requiere que una parte del sistema tenga la misma funcionalidad no sea necesario duplicar la porción de código con la que se realiza.
Confiable	El sistema debe garantizar la integridad de datos.

Elaborado por: Paul Jara & Javier Rivera.

2.2.2 Restricciones.

Tabla 4. Restricciones de la arquitectura.

Restricción	Requerimiento arquitectónico.
Desarrollo	El sistema debe ser escrito en Java.
Costo	Las herramientas utilizadas para la realización del sistema deben ser libres.

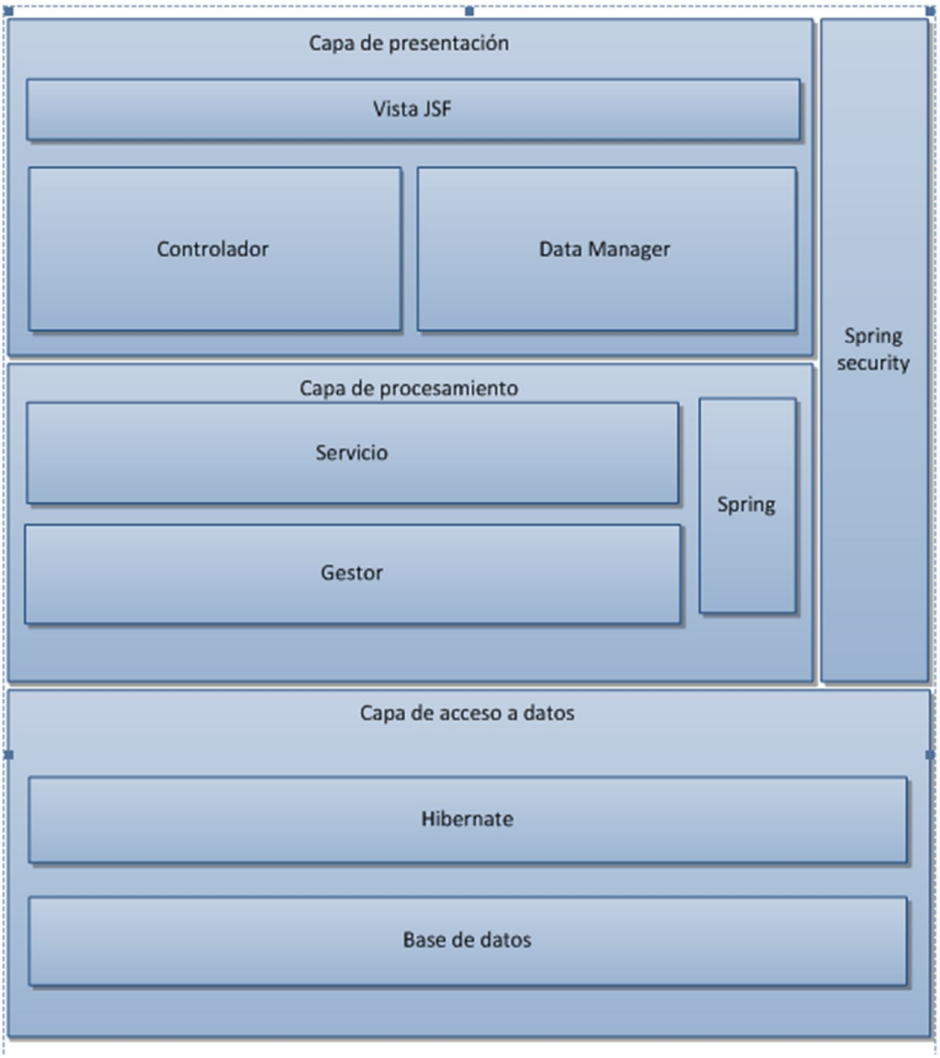
Elaborado por: Paul Jara & Javier Rivera.

2.2.3 Arquitectura N capas.

Una arquitectura N capas divide al sistema en distintas unidades funcionales: cliente, presentación, lógica de negocio, y sistema de información empresarial. Esto asegura una división de responsabilidades y hace que el sistema sea fácilmente mantenible y extensible. Los sistemas con N capas son más escalables y flexibles que un sistema cliente-servidor, en el que no existe una capa de lógica de negocio separada como tal.

Las capas de la aplicación interactúan entre ellas de manera que cada una se comunica con la capa que se encuentra adyacente, esto simplifica el mantenimiento de la aplicación y la implementación de nuevos requerimientos.

Figura 9. Diseño de la arquitectura para el sistema web



Elaborado por: Paul Jara & Javier Rivera.

2.2.4 La capa de presentación.

Esta capa expone los servicios de la capa de procesamiento a los usuarios, se encarga de la interacción y el procesamiento de las peticiones del cliente. Para la aplicación web, en esta capa se utiliza la arquitectura MVC con JSF.

MVC es el patrón de diseño arquitectural recomendado para aplicaciones interactivas Java, este patrón separa los conceptos de diseño, por lo tanto ayuda a la

reutilización de código, centraliza el control y hace que la aplicación sea más extensible, también ayuda a los desarrolladores a enfocarse en sus habilidades principales y a colaborar a través de interfaces claramente definidos.

- **Modelo:** son los beans que contienen la información que se va a presentar en la página y contienen funcionalidad de la vista.
- **Vista:** son las páginas JSF con extensión XHTML, en estas se encuentran los componentes con los que el usuario interactúa.
- **Controlador:** son clases Java desde las cuales se realizan las peticiones a los servicios de la aplicación. El controlador también contiene la manera en que el usuario navega a través de las funcionalidades del sistema web.

JSF (JavaServer Faces) es un framework de componentes de interface (UI) para aplicaciones web basadas en Java, contiene un API para representar componentes y manejar sus estados, eventos, validación del lado del servidor, conversión de datos, definir la navegación entre páginas, etc.

JSF encaja bien en la arquitectura de la capa de presentación basada en MVC ya que ofrece una clara separación entre el comportamiento y la presentación, asocia a cada vista con un conjunto de objetos Java manejados por el controlador (managed beans), estos objetos facilitan la recolección, manipulación y visualización de los valores mostrados en los diferentes elementos de los formularios, sus principales funcionalidades son:

- **Desarrollo orientado a objetos al estilo swing**.

El modelo de componentes UI con estado del lado del servidor permiten manejar la lógica de presentación del sistema web modularizada.

- **Control de backing beans.**

Los backing beans son componentes JavaBeans asociados con componentes UI utilizados en la página, separa la definición de los componentes del UI de los objetos que realizan el procesamiento específico de la aplicación.

- **Modelo de componentes UI extensible.**

Los componentes UI de JSF son elementos configurables, reutilizables que componen los interfaces de usuario de aplicaciones JSF, se utilizan para crear los componentes personalizados de la aplicación web, se puede extender de un componente UI estándar para desarrollar componentes más complejos que incluyan validaciones u otro comportamiento requerido.

- **Modelo de renderizado flexible**.

Un renderizador separa la vista y la funcionalidad de los componentes UI, se pueden crear y utilizar varios renderizadores para definir diferentes apariencias del mismo componente para distintos usuarios o diferentes páginas en donde se utilice el componente.

- **Modelo de conversión y validación extensible.**

Basados en los convertidores y validadores estándar, se pueden desarrollar convertidores y validadores personalizados, que proporcionan la integridad de datos que se requiere antes de que estos sean procesados por la capa de lógica de negocio.

2.2.5 La capa de procesamiento.

Es la capa de la lógica de negocio, contiene los objetos, servicios y gestores de negocio de la aplicación, recibe peticiones de la capa de presentación y se encarga de procesar la lógica de negocio, además maneja los accesos a los recursos del sistema de información empresarial. Los componentes de la capa de lógica de negocio se

benefician de la mayoría de los servicios de nivel transversal como el control de seguridad, de transacciones y de recursos.

2.2.5.1 Servicios.

Los servicios se encargan de exponer la lógica de negocio hacia la capa de presentación por medio de una interface, son manejados por el framework Spring, mediante inyección de dependencias, para esto utilizan anotaciones que los identifican como beans de Spring, además tienen la función de abrir una transacción en la sesión de Hibernate de manera que si se presenta un error durante la ejecución de esta, las operaciones realizadas contra la base de datos realicen un rollback, para mantener la integridad de datos y la atomicidad de la transacción.

2.2.5.2 Gestores.

Los gestores son objetos manejados por el framework Spring que mediante inyección de dependencias obtienen acceso a otros gestores y estos a su vez son inyectados en los servicios, contienen la lógica de negocio donde se realizan operaciones sobre los datos obtenidos y cálculos para procesar la información de la aplicación.

2.2.6 La capa de acceso a datos.

Los datos de la aplicación persisten en la capa de información empresarial que incluye bases de datos relacionales, bases de datos orientadas a objetos, y sistemas antiguos. En esta capa se emplea Hibernate como ORM por las facilidades para recuperación y actualización de datos, control de transacciones, repositorio de conexiones a bases de datos, consultas programáticas y declarativas, y control de relaciones de entidades declarativas.

Permite desarrollar objetos persistentes siguiendo el lenguaje común de Java incluyendo asociación, herencia, polimorfismo, composición y el marco de trabajo Collections para obtener los datos de las tablas relacionadas.

2.2.7 Seguridad.

La seguridad es un tema fundamental en todos los sistemas de información, por este motivo se utiliza un utilitario que se encuentra perpendicular a las capas del sistema afectando a cada una de ellas. Spring Security es un subproyecto del framework Spring, que permite gestionar completamente la seguridad de la aplicación, ofrece algunas características interesantes como el poder separar la lógica de la aplicación del control de la seguridad, gestiona la seguridad en diferentes niveles: URLs, métodos o instancias concretas, es portable debido a se encuentra dentro del WAR o el EAR de la aplicación.

2.3 Diagramas de procesos

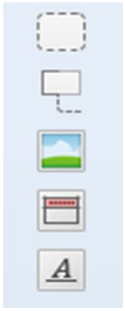
Mediante el diagrama de procesos se determina las actividades a desarrollar por un proceso para cumplir con un objetivo. Para la elaboración del diagrama de procesos se empleó la notación BPMN (Business Process Modeling Notation), un estándar internacional de modelado, que describe la lógica paso a paso de un proceso de negocio. BPMN ofrece algunas ventajas:

- Permite modelar procesos de manera unificada y estandarizada, para facilitar la comprensión a todas las personas de una organización.

- Crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.

2.3.1 Simbología.

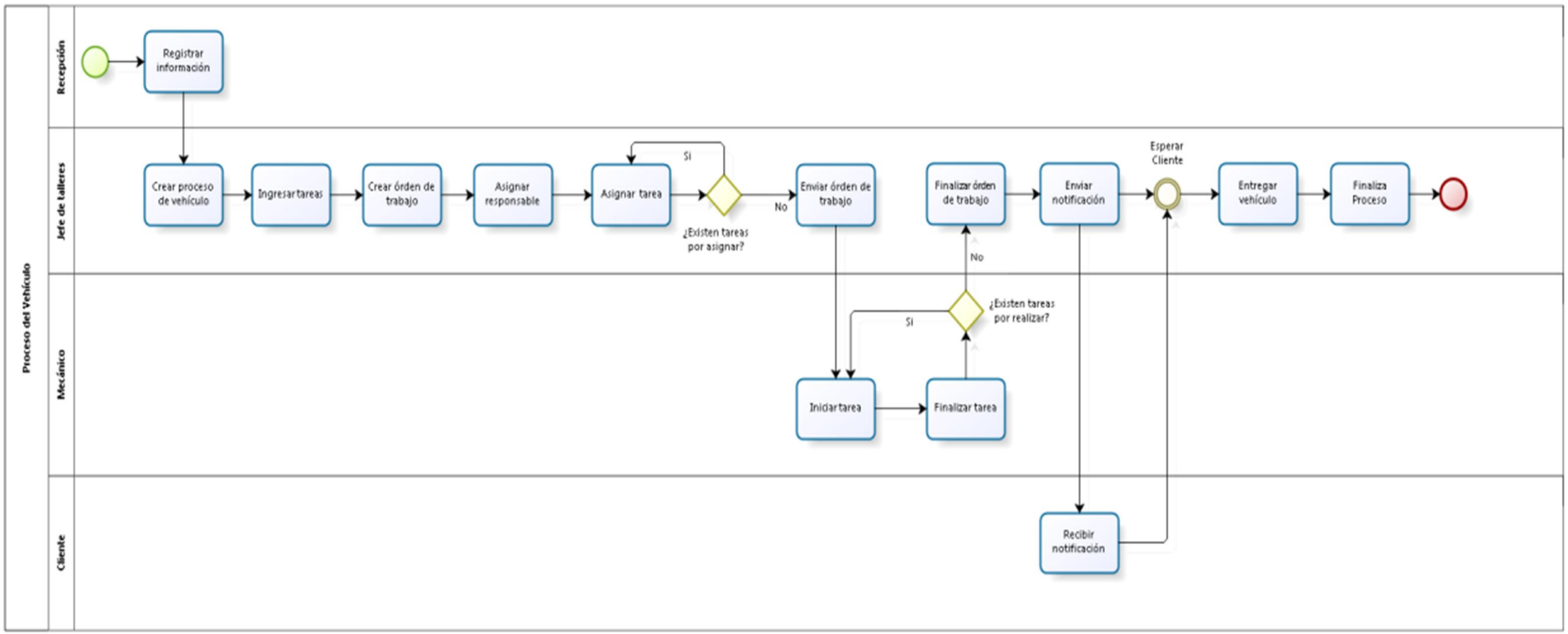
Tabla 5. Simbología de BPMN

Categoría	Objeto	Descripción	Ejemplo
Objetos de flujo	Evento	Es algo que sucede dentro de un proceso de negocio y afectan al flujo del proceso	
	Actividades	Es un término genérico para un trabajo ejecutado.	
	Gateway entradas	Representa decisiones, bifurcaciones de las fusiones o uniones dentro del diagrama	
Objetos de conexión	Flujo de secuencia	Se usan para mostrar el orden de los eventos	
	Flujo de mensajes	Se usa para indicar el flujo de los mensajes entre las entidades.	
	Flujo de asociación	Usados para asociar diferentes artefactos de objeto del flujo	
Swimlanes	Piscinas	Identifican los participantes dentro de un flujo de trabajo	
	Carriles	Están dentro de la piscina indican quien realiza que dentro del proceso.	
Artefactos	Objetos de datos	Es un mecanismo para mostrar como los datos son requeridos y producidos por las actividades	
	Grupo	Se usa para finalidades de documentación o de análisis	
	Anotaciones	Son mecanismos para incluir información adicional para el lector del diagrama.	

Fuente:slideshare.net

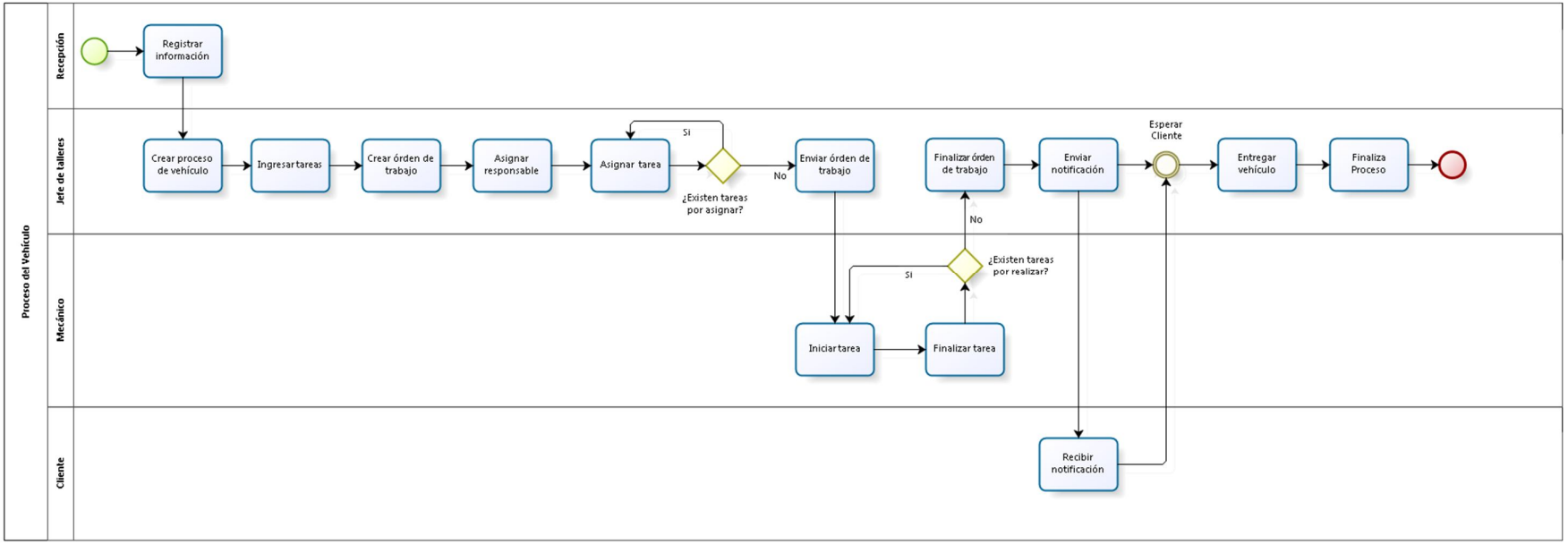
Los diagramas de los principales procesos del sistema se detallan a continuación:

Figura 10. Diagrama de proceso del vehículo



Elaborado por: Paul Jara & Javier Rivera.

Figura 11. Diagrama del proceso de notificaciones



Elaborado por: Paul Jara & Javier Rivera.

2.4 Diagramas UML

2.4.1 Modelo de casos de uso.

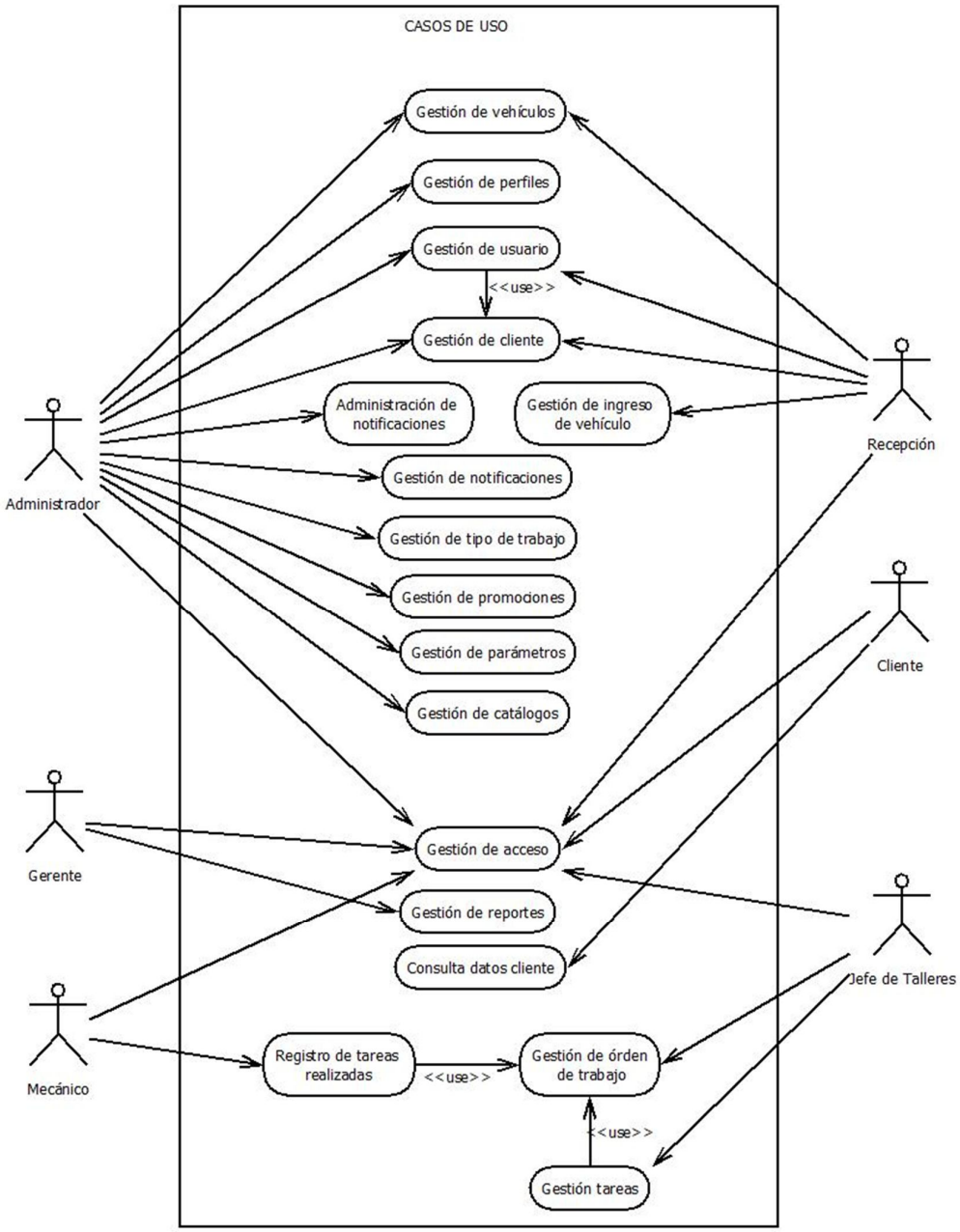
Se han considerado los siguientes actores para el sistema.

Tabla 6. Actores del sistema.

Núm.	Actor	Descripción
1	Administrador	Este usuario se encarga de gestionar los datos del sistema, como la configuración de parámetros y asignación de permisos.
2	Mecánico	Este usuario se encarga de ejecutar las órdenes de trabajo y llevar a cabo el seguimiento de los procesos del cliente.
3	Gerente	Este usuario se encarga de generar y analizar los reportes del sistema en función a la productividad y desempeño de la empresa.
4	Cliente	Este usuario visualiza la información del vehículo, notificaciones y promociones, además puede enviar solicitudes para agendar citas.
5	Jefe de Talleres	Este usuario se encarga de la gestionar de las órdenes de trabajo.
6	Recepción	Este usuario se encarga ingresar la información de vehículos y clientes atendidos en la mecánica.

Elaborado por: Paul Jara & Javier Rivera.

Figura 12. Casos de uso del sistema web



Elaborado por: Paul Jara & Javier Rivera..

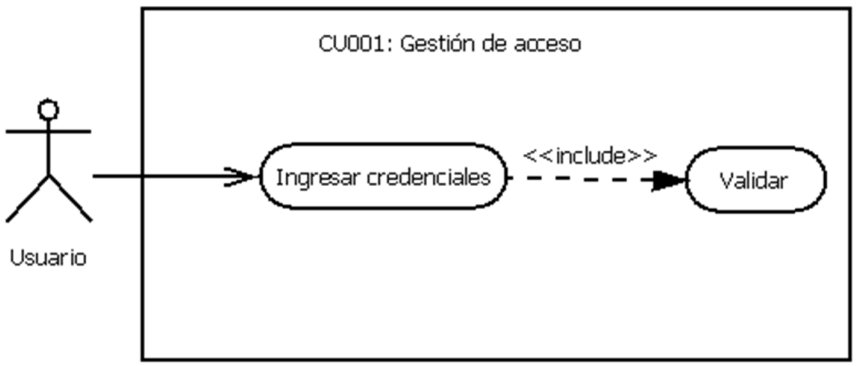
2.4.1.1 Caso de uso gestión de acceso.

Tabla 7. Especificación del caso de uso gestión de acceso

Código	CU001
Nombre	Gestión de acceso.
Tipo	Primario
Descripción	En este caso de uso se registrarán los eventos que suceden al efectuar el proceso de autenticado en el sistema.
Objetivo	Registrar y validar el ingreso de un usuario al sistema.
Actores	Todos los usuarios.
Precondiciones	• El usuario deberá estar registrado en el sistema. • El usuario deberá tener un perfil activo en el sistema. • El perfil deberá tener acceso a las funcionalidades del sistema. • El usuario deberá acceder a la página de la dirección principal del sistema.
Post condiciones de éxito	• El usuario podrá interactuar con las diferentes funcionalidades que ofrece la aplicación de acuerdo a los permisos de los que disponga el perfil asignado.
Post condiciones de falla	• Se indicará el motivo por el cual no se realiza la autenticación del usuario. • Se mantendrá en la pantalla de autenticación.
Flujo normal	a) El flujo principal comenzará cuando el usuario acceda mediante el navegador a la dirección del sistema. b) Se desplegará una pantalla con los campos habilitados para la autenticación en el sistema. c) El usuario deberá ingresar la siguiente información: nombre de usuario y contraseña. d) El usuario presionará el botón ***Ingresar*** para ejecutar el proceso de autenticación. e) Se desplegará la página de inicio del sistema.
Flujos alternativos	a) En el paso (d) del flujo normal: Sí el usuario no ingresa alguno de los datos requeridos para la autenticación se deberá indicar que el dato es requerido. b) Sí se presentase algún error interno durante la ejecución se presentará un mensaje que indica el error durante la autenticación.

Elaborado por: Paul Jara & Javier Rivera.

Figura 13. Diagrama del caso de uso gestión de acceso



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.2 Caso de uso gestión de usuarios.

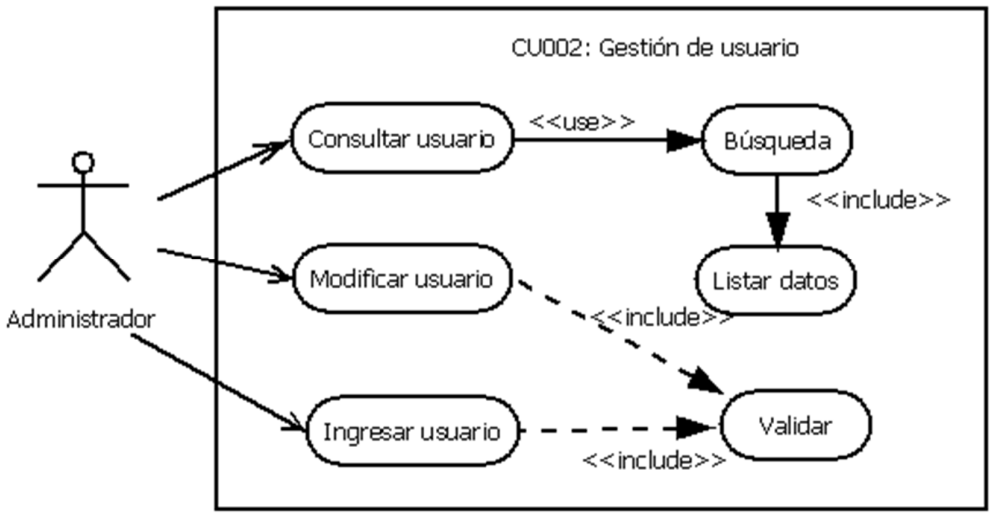
Tabla 8. Especificación del caso de uso gestión de usuarios

Código	CU002
Nombre	Gestión de usuarios.
Descripción	Este caso de uso mencionará los eventos que se presentan al registrar, modificar o inactivar un usuario del sistema.
Objetivo	Registrar, modificar o inactivar un usuario del sistema
Actores	Administrador
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el rol de administrador.
Post condiciones de éxito	• El usuario será creado o modificado. • Sí es nuevo pasará a estado activo. • Se podrá utilizar el usuario dentro del sistema para las diferentes funcionalidades.
Post condiciones de falla	• El usuario interno no será creado o modificado. • Se indicará el motivo por el cual no se pudo realizar el proceso.
Flujo normal	a) El flujo principal comenzará cuando el actor ingrese a la opción administración de usuarios. b) Se desplegará una pantalla con los usuarios ingresados. c) El actor elegirá entre las opciones ***Ingresar*** o ***Editar*** para cargar el formulario con los datos del usuario.

	d) El actor deberá ingresar la siguiente información: nombre de usuario, contraseña, correo electrónico además deberá elegir el/los tipo(s) de perfil y el tipo de identificación de persona. e) El actor registrará el ingreso de los datos presionando el botón **Guardar**, donde se realizarán las validaciones necesarias para el proceso. f) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso (a) del flujo normal: El actor podrá realizar una búsqueda filtrando con el nombre de usuario. b) En el paso (d) del flujo normal: Se podrá inactivar el usuario cambiando el valor del campo activo. c) En el paso (e) del flujo normal: Si en la creación o modificación del usuario se presentase algún error se desplegará una notificación.

Elaborado por: Paul Jara & Javier Rivera.

Figura 14. Diagrama del caso de uso gestión de usuarios



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.3 Caso de uso gestión de datos de cliente.

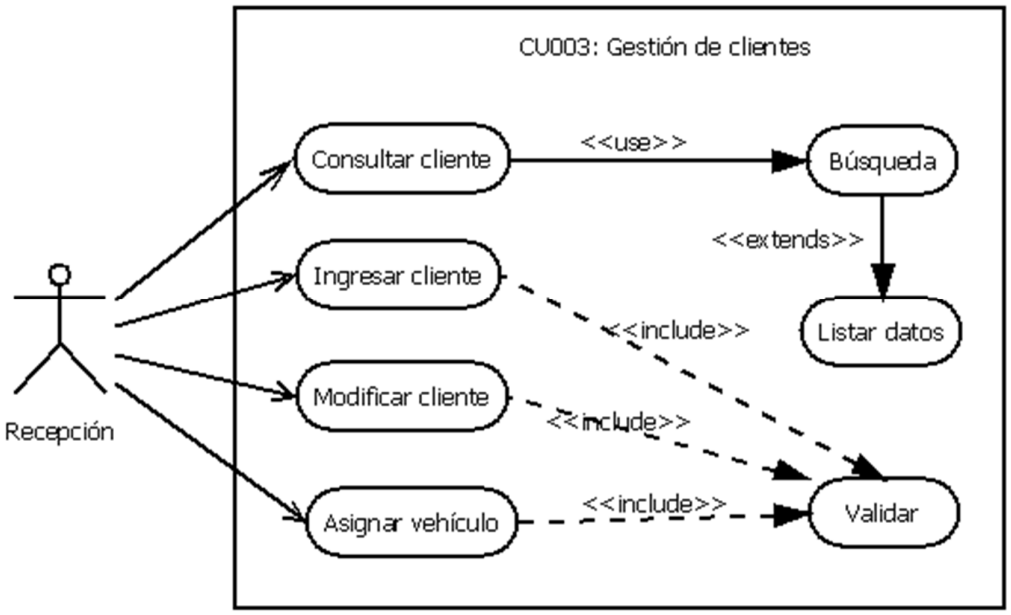
Tabla 9. Especificación del caso de uso gestión de datos de cliente

Código	CU003
Nombre	Gestión de datos de cliente

Descripción	En este caso de uso se detallará los eventos que se presentan en el proceso de registro, modificación y eliminación de los datos del cliente incluyendo la asignación de vehículo.
Objetivo	Registrar, consultar e inactivar información del cliente, necesaria para las funcionalidades del sistema.
Actores	Recepción
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil de recepción.
Post condiciones de éxito	• Los datos del cliente serán ingresados. • Se podrá utilizar al cliente en las funcionalidades del sistema.
Post condiciones de falla	• Los datos del cliente no serán ingresados. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) Se desplegará una búsqueda de clientes ingresados en el sistema, donde el actor podrá elegir un registro para editarlo. b) El actor elegirá entre las opciones ***Ingresar*** o ***Editar*** para cargar el formulario con los datos del cliente. c) El actor deberá ingresar la siguiente información: identificación, nombres, apellidos, dirección, teléfono(s), correo electrónico. d) El actor registrará los datos del usuario presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. e) El actor recibirá una respuesta sobre el proceso. f) EL actor podrá elegir la opción ***Asignar vehículo***, donde se desplegará una búsqueda de vehículos ingresados en el sistema. g) El actor seleccionará uno o varios vehículos y guarda. h) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso (a) del flujo normal: El actor podrá realizar una búsqueda de acuerdo a la identificación o al nombre de la persona. b) En el paso (d) del flujo normal: i. Se podrá inactivar al cliente cambiando el valor del campo activo. ii. El actor podrá terminar sin guardar mediante el botón ***Cancelar***. c) En el paso (e) del flujo normal: Sí en la creación o modificación del cliente se presentase algún error se desplegará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 15. Diagrama del caso de uso gestión de datos de cliente



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.4 Caso de uso gestión de vehículos.

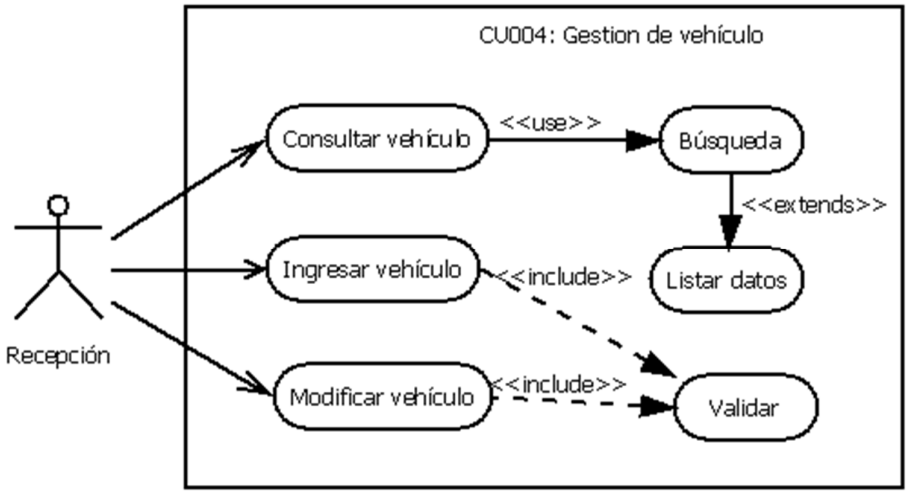
Tabla 10. Especificación caso de uso gestión de vehículo

Código	CU004
Nombre	Gestión de vehículos
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de registro de datos del vehículo.
Objetivo	Registrar, consultar e inactivar la información del vehículo necesaria para las funcionalidades del sistema.
Actores	Administrador Mecánico
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener perfil mecánico. • El actor deberá acceder a la sección gestión de vehículos.
Post condiciones de éxito	• Los datos del vehículo serán ingresados o modificados. • Se podrá utilizar la información ingresada en las funcionalidades del sistema.

Post condiciones de falla	• Los datos del vehículo no serán registrados. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) Se desplegará una pantalla con un control para la búsqueda de vehículos, donde mediante el ingreso de la placa se puede acceder a la información, en caso de no existir se desplegará un mensaje y se presentarán los controles para realizar el ingreso de un nuevo vehículo. b) El actor deberá ingresar la siguiente información: placa, modelo, año, número de motor, chasis y color del vehículo, además tendrá que elegir el tipo de vehículo. c) El actor guardará los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. d) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso (d) del flujo normal: i. Se podrá inactivar el vehículo cambiando el valor del campo activo. ii. El actor podrá terminar sin guardar mediante el botón ***Cancelar***. b) En el paso (e) del flujo normal: Sí en la creación o modificación del vehículo se presentase algún error se desplegará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 16. Diagrama del caso de uso de gestión de vehículos



Elaborado por: Paul Jara & Javier Rivera.

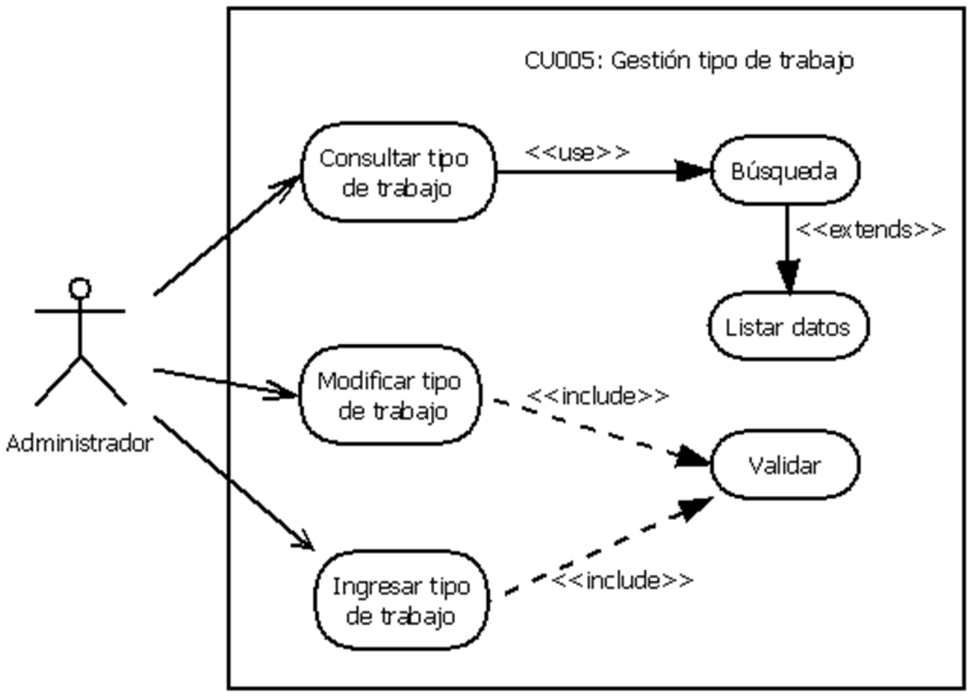
2.4.1.5 Caso de uso gestión de tipo de trabajo.

Tabla 11. Especificación del caso de uso gestión de tipo de trabajo.

Código	CU005
Nombre	Gestión de tipo de trabajo.
Descripción	En este documento se detallarán los eventos que se presentan en el proceso de registro, modificación y desactivación de los tipos de tareas que realizan los mecánicos.
Objetivo	Registrar, consultar y desactivar el tipo de tarea.
Actores	Administrador
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el rol de administrador. • El actor deberá acceder a la sección gestión de tipos de tareas.
Post condiciones de éxito	• Los datos del tipo de tarea serán registrados. • Se podrá modificar los datos del tipo de tarea ingresados.
Post condiciones de falla	• Los datos del tipo de tarea no serán ingresados. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) Se desplegará una pantalla con los tipos de tareas ingresados en el sistema, donde el actor podrá elegir un registro para ser editado. b) El actor elegirá entre las opciones ***Ingresar*** o ***Editar***. c) El actor deberá ingresar la siguiente información: código, nombre, descripción. d) El actor registrará el ingreso de los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. e) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso a) del flujo normal: El actor podrá realizar una búsqueda de acuerdo al código. b) En el paso d) del flujo normal: i. Se podrá inactivar el tipo de tarea cambiando el valor del campo activo. ii. El actor podrá terminar sin guardar mediante un botón ***Cancelar***. c) En el paso e) del flujo normal: Sí en la creación o modificación del tipo de tarea se presentase algún error se desplegará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 17. Diagrama del caso de uso gestión de tipo de trabajo



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.6 Caso de uso gestión de ingreso de vehículo.

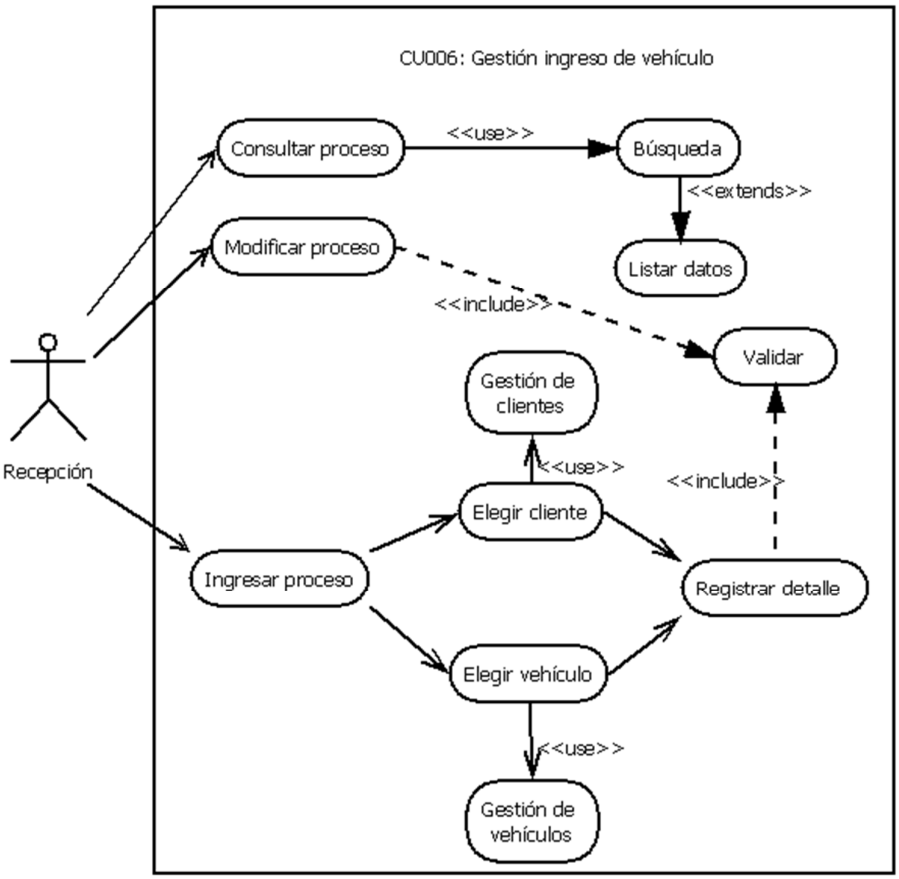
Tabla 12. Especificación caso de uso gestión de ingreso de vehículo.

Código	CU006
Nombre	Gestión de ingreso de vehículo.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de registro de la ficha de ingreso del vehículo a los talleres.
Objetivo	Registrar la información necesaria del ingreso de un vehículo a talleres.
Actores	Recepción Jefe de talleres
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el rol mecánico. • El actor deberá acceder a la sección fichas de ingreso.

Post condiciones de éxito	• El registro será guardado. • Se podrá utilizar esta información para la elaboración de la ficha de trabajo.
Post condiciones de falla	• La ficha de ingreso no será registrada. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) De entrada se desplegará una tabla con las fichas ingresadas que aún no tienen una orden de trabajo. b) El actor podrá crear una nueva ficha presionando el botón ***Nuevo*** o modificar una existente mediante el botón ***Editar***. c) El actor deberá elegir la siguiente información de la ficha: identificación del cliente, número de vehículo, fecha del proceso. d) El actor podrá salvar la información del proceso mediante el botón ***Guardar***. e) El actor recibe una respuesta con el resultado del proceso.
Flujos alternativos	a) En el paso (c) del flujo normal: i. El actor deberá elegir de un buscador el cliente, de no existir mediante un link se dirige al proceso de gestión de cliente. ii. El actor deberá elegir de un buscador el vehículo, de no existir mediante un link se dirige al proceso de gestión de vehículo. b) En el paso (d) del flujo normal: El actor podrá regresar a la pantalla anterior donde podrá corregir los errores o su vez terminar mediante un botón ***Cancelar***. c) En el paso (e) del flujo normal: d) Si en la creación o modificación de la ficha de ingreso se presentase algún error será necesario desplegar una notificación.

Elaborado por: Paul Jara & Javier Rivera

Figura 18. Diagrama del caso de uso gestión de ingreso de vehículo



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.7 Caso de uso gestión de orden de trabajo.

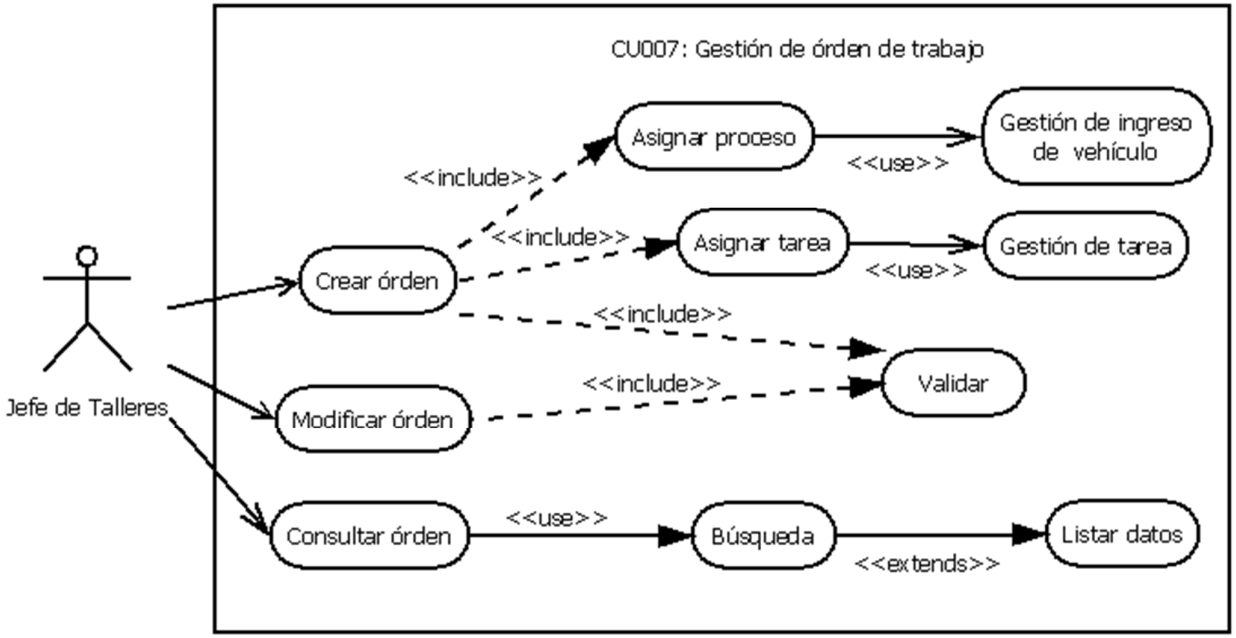
Tabla 13. Especificación del caso de uso gestión de orden de trabajo.

Código	CU007
Nombre	Gestión de orden de trabajo.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de elaboración de las órdenes de trabajo que serán usadas por los mecánicos para el desarrollo del trabajo.
Objetivo	Registrar, consultar y modificar los datos de las órdenes de trabajo.
Actores	Jefe de talleres
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil Jefe de mecánicos.

	• Deberá existir una ficha de ingreso registrada.
Post condiciones de éxito	• Los datos de la orden de trabajo serán ingresados. • Se podrá modificar los datos de la orden ingresados. • Se podrá imprimir la orden de trabajo para ser efectuada.
Post condiciones de falla	• Los datos de la orden de trabajo no serán ingresados. • Se indicará el motivo por el cual no se pudo completar el proceso satisfactoriamente.
Flujo normal	a) De inicio se desplegará una tabla con las órdenes de trabajo ingresadas. Donde el actor podrá elegir un registro para ser editado. b) El actor elegirá entre las opciones ***Ingresar*** o ***Editar***. c) El actor ingresará o modificará los datos requeridos para la elaboración de las órdenes de trabajo. d) El actor deberá elegir la ficha de ingreso a ser desarrollada, el mecánico a cargo, y al menos una tarea a ser efectuada. La orden de trabajo se establece como estado inicial *Ingresada*. e) El usuario recibirá una respuesta con el resultado del proceso.
Flujos alternativos	a) En el paso a) del flujo normal: El actor podrá realizar una búsqueda de acuerdo al número automático de orden de trabajo. b) En el paso d) del flujo normal: El actor podrá terminar sin guardar mediante un botón ***Cancelar***. c) En el paso e) del flujo normal: Sí en la creación o modificación de la orden de trabajo se presentase algún error se desplegará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 19. Diagrama del caso de uso gestión de orden de ingreso



Elaborado por: Paul Jara & Javier Rivera.

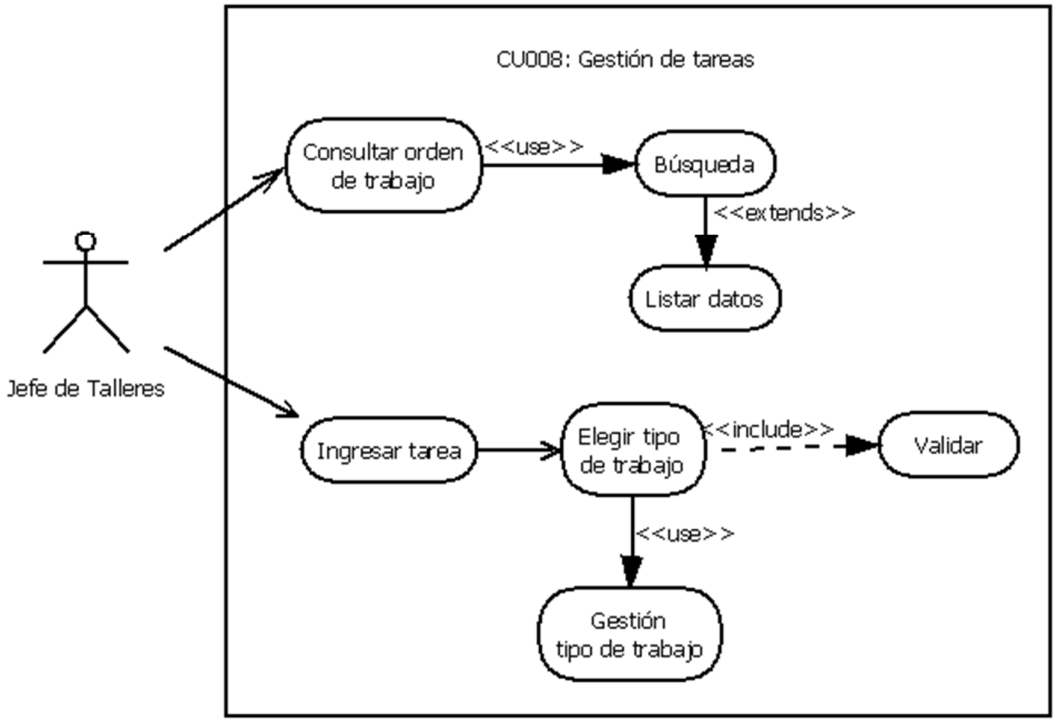
2.4.1.8 Caso de uso gestión de tareas.

Tabla 14. Especificación del caso de uso gestión de tareas.

Código	CU008
Nombre	Gestión de tareas.
Descripción	En este documento se detallarán los eventos que se presentan en el proceso de asignación, modificación y desactivación de tareas en la orden de trabajo.
Objetivo	Asignar, consultar o desactivar los las tareas en la orden de trabajo.
Actores	Jefe de Talleres
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el rol de jefe de talleres. • Deberá existir al menos un tipo de trabajo ingresado.
Post condiciones de éxito	• Los datos de la tarea serán registrados o modificados. • Se podrá asignar las tareas activas a la orden de trabajo.
Post condiciones de falla	• Los datos de la tarea no serán registrados. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) El sistema deberá presentar una búsqueda de órdenes ingresadas, el actor podrá elegir un registro para asignar o modificar las tareas. b) Se desplegará una pantalla con las Tareas Ingresadas en el sistema, donde el actor podrá elegir un registro para ser asignado. c) El actor registrará el ingreso de los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. d) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso b) del flujo normal: i. Se podrá inactivar la tarea asignada cambiando el valor del campo *activo*. ii. El actor podrá terminar sin guardar mediante un botón ***Cancelar***. b) En el paso e) del flujo normal: Sí en la creación o modificación de la tarea se presentase algún error se desplegará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 20. Diagrama del caso de uso gestión de tareas



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.9 Caso de uso gestión de perfil.

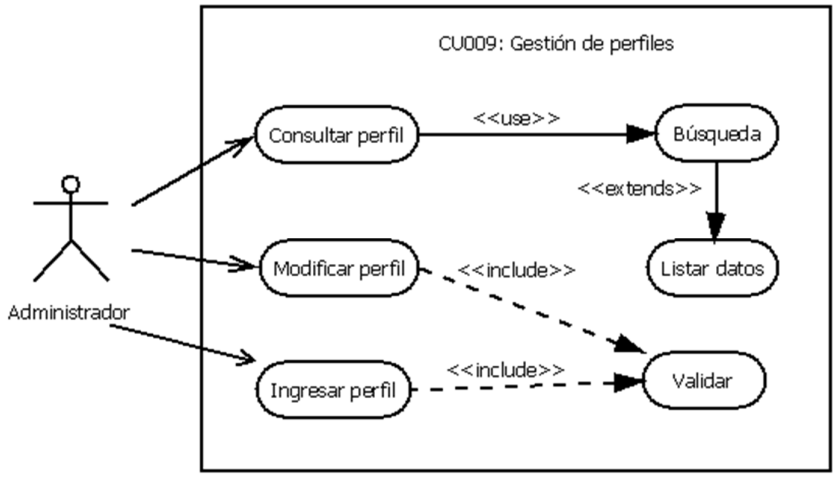
Tabla 15. Especificación del caso de uso gestión de perfil.

Código	CU009
Nombre	Gestión de perfil.
Descripción	Este caso de uso mencionará los eventos que se presentan al registrar, modificar o inactivar un perfil de usuario.
Objetivo	Registrar, modificar o inactivar un perfil de usuario.
Actores	Administrador
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el rol de administrador.

Post condiciones de éxito	• El perfil de usuario será creado o modificado. • El perfil de usuario interno si es nuevo pasará a un estado *Activo*.
Post condiciones de falla	• El perfil de usuario no será creado o modificado. • Se indicará el motivo por el cual no se pudo realizar el proceso.
Flujo normal	a) El flujo principal comenzará cuando el actor ingresa a la opción gestión de perfiles. b) Se desplegará una pantalla con los perfiles de usuario ingresados. c) El actor elegirá entre las opciones ***Ingresar*** o ***Editar*** para cargar el formulario con los datos del perfil. d) El actor deberá ingresar la siguiente información: código, nombre. e) El actor registrará el ingreso de los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. f) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso (a) del flujo normal: El actor podrá realizar una búsqueda de acuerdo al código del perfil. b) En el paso (d) del flujo normal: i. Se podrá inactivar el perfil cambiando el valor del campo ***activo***. ii. El usuario podrá regresar a la pantalla anterior donde podrá corregir los errores o su vez terminar mediante un botón ***Cancelar***. c) En el paso (e) del flujo normal: Si en la creación o modificación del perfil se presentase algún error es necesario realizar una notificación.

Elaborado por: Paul Jara & Javier Rivera.

Figura 21. Diagrama del caso de uso gestión de perfil



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.10 Caso de uso registro de tareas realizadas.

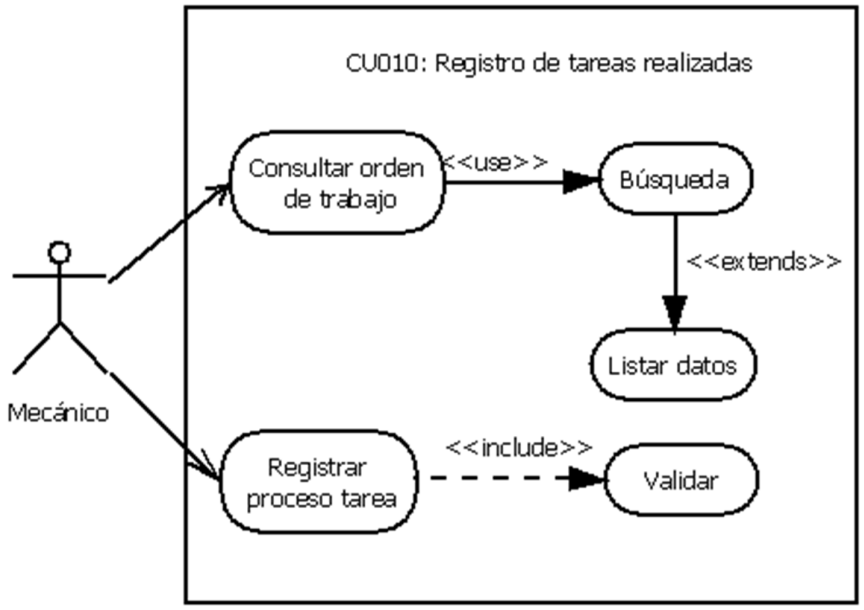
Tabla 16. Especificación del caso de uso registro de tareas realizadas.

Código	CU010
Nombre	Registro de tareas realizadas.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de registrar los avances sobre las tareas expuestas en la orden de trabajo.
Objetivo	Registrar, consultar y modificar las tareas realizadas de las órdenes de trabajo.
Actores	Mecánico
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil mecánico. • Deberá existir una orden de trabajo ingresada.
Post condiciones de éxito	• Los datos de tareas realizadas serán registrados. • Se podrá evaluar el avance de la orden de trabajo para información del cliente.
Post condiciones de falla	• Los datos de las tareas realizadas no serán ingresados. • Se indicará el motivo por el cual no se pudo guardar.

Flujo normal	a) De inicio se desplegará una tabla con las órdenes de trabajo ingresadas. b) El actor elegirá la orden sobre la cual se encuentra trabajando, el sistema despliega un formulario con la información de la orden de trabajo y sus tareas c) El actor ingresará o modificará el porcentaje de avance sobre la tarea. d) El actor salvará la información mediante el botón ***Guardar***. e) El usuario recibirá una respuesta con el resultado del proceso.
Flujos alternativos	a) En el paso a) del flujo normal: El actor podrá realizar una búsqueda de acuerdo al número automático de orden de trabajo. b) En el paso d) del flujo normal: El actor podrá terminar sin guardar mediante un botón ***Cancelar.*** c) En el paso e) del flujo normal: Si en el registro de creación o modificación de la orden de trabajo se presentase algún error será necesario realizar una notificación y regresar a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 22. Diagrama del caso de uso registro tareas realizadas



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.11 Caso de uso gestión notificaciones.

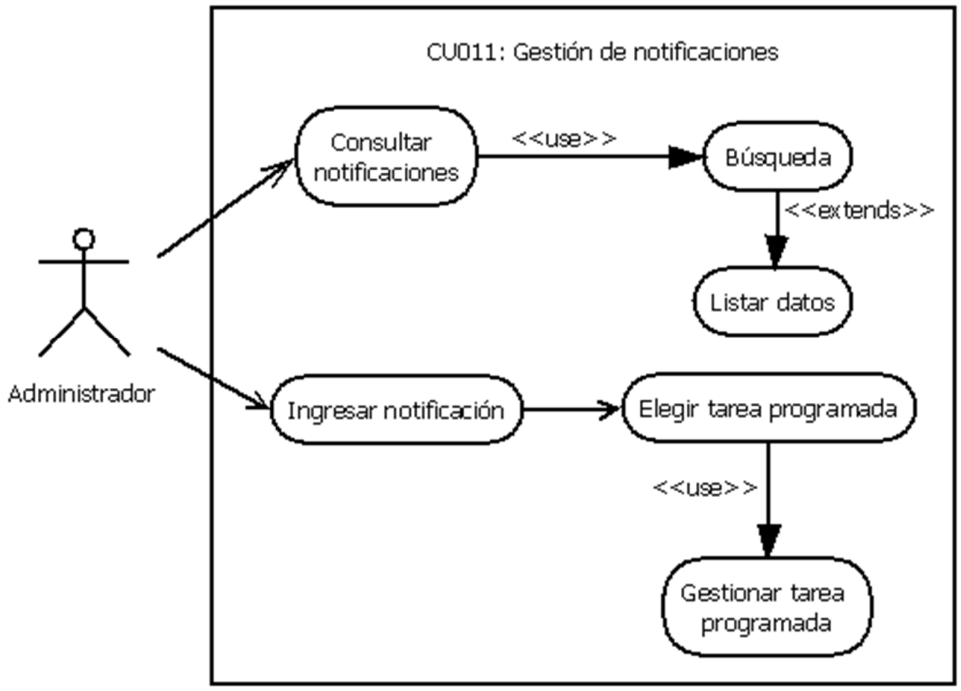
Tabla 17. Especificación del caso de uso gestión notificaciones

Código	CU011
Nombre	Gestión notificaciones.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de crear modificar y eliminar notificaciones a clientes y la asignación de tareas programadas para su ejecución
Objetivo	Consultar, Crear, modificar, eliminar notificaciones y tareas programadas a clientes.
Actores	Administrador
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil administrador.
Post condiciones de éxito	• Las notificaciones a los clientes serán ingresadas o modificadas.
Post condiciones de falla	• La notificación no será ingresada.
Flujo normal	a) Se desplegará una pantalla con una búsqueda de notificaciones ingresados en el sistema, donde el actor podrá elegir un registro para ser editado. b) El actor elegirá entre las opciones ***Ingresar*** o ***Editar***. c) El actor deberá ingresar la siguiente información: nombre, cuerpo, requiere respuesta, activo. d) El actor registrará el ingreso de los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. e) El actor recibirá una respuesta sobre el proceso. f) El sistema le desplegará la opción agregar tarea programada. g) El actor ingresará el tipo de tarea (periódica, inmediata). Si es inmediata procede al envió, caso contrario solicita la fecha a ser enviada. h) El actor salva los cambios.

Flujos alternativos	a) En el paso d) del flujo normal: El actor podrá terminar sin guardar mediante un botón ***Cancelar***. b) En el paso e) del flujo normal: Si en el registro de creación o modificación de notificaciones se presentase algún error será necesario realizar una notificación y regresar a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 23. Diagrama del caso de uso gestión notificaciones



Elaborado por: Paul Jara & Javier Rivera.

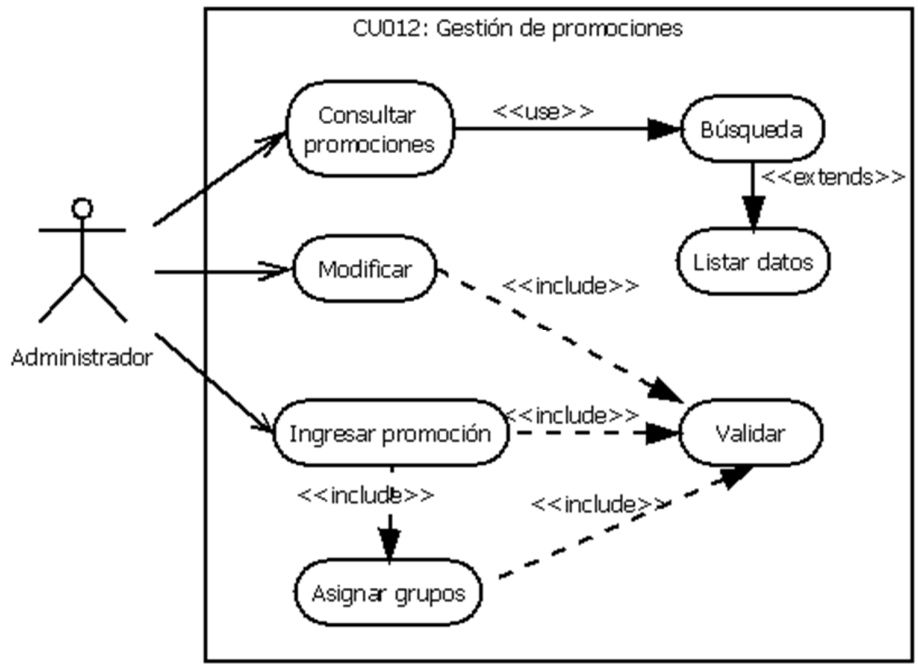
2.4.1.12 Caso de uso gestión de promociones.

Tabla 18. Especificación del caso de uso gestión de promociones.

Código	CU012
Nombre	Gestión de promociones.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de crear, modificar e inactivar promociones además de la asignación a grupos destinatarios.
Objetivo	Administrar las promociones a ser presentadas tanto en el sistema web como en el dispositivo móvil.
Actores	Administrador
Precondiciones	El actor deberá estar autenticado en el sistema. El actor deberá tener el perfil administrador.
Post condiciones de éxito	La promoción será ingresada o modificada.
Post condiciones de falla	La promoción no será ingresada o modificada.
Flujo normal	a) Se desplegará una pantalla con una búsqueda de promociones ingresadas en el sistema, donde el actor podrá elegir un registro para ser editado. b) El actor elije entre las opciones ***Ingresar*** o ***Editar***. c) El actor deberá ingresar la siguiente información: descripción, cuerpo, fecha de inicio y fin de vigencia, activo. d) El actor registrará el ingreso de los datos presionando el botón ***Guardar***, donde el sistema realizará las validaciones necesarias para el proceso. e) El actor recibirá una respuesta sobre el proceso. f) El sistema le desplegará la opción agregar o modificar los grupos asignados. g) El sistema desplegará los grupos ingresados con un check para la elección. h) El actor salva los cambios.
Flujos alternativos	a) En el paso d) del flujo normal: El actor podrá terminar sin guardar mediante un botón ***Cancelar***. b) En el paso e) del flujo normal: Si en el proceso se presentase algún error, se realizará una notificación y regresará a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 24. Diagrama del caso de uso gestión de promociones



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.13 Caso de uso administración de parámetros.

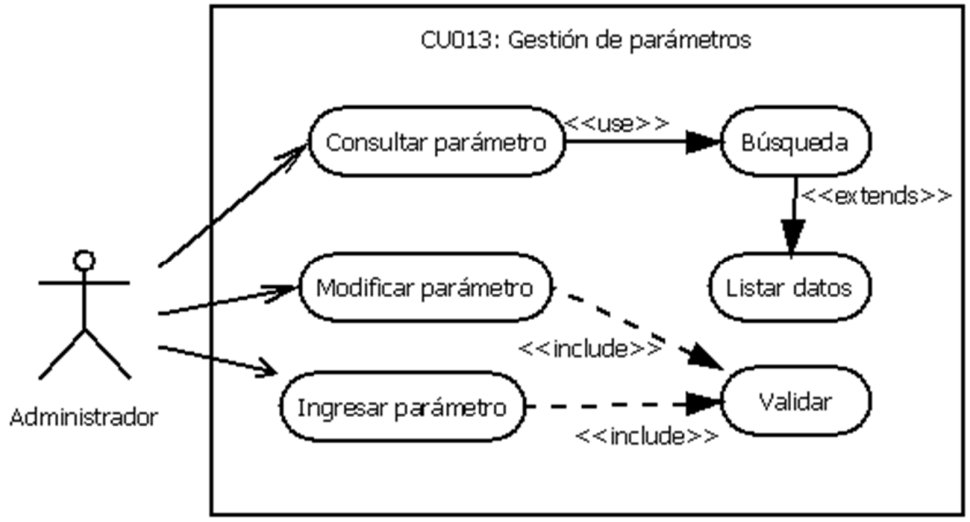
Tabla 19. Especificación del caso de uso administración de parámetros.

Código	CU013
Nombre	Administración de parámetros.
Descripción	En este caso de uso se detallará los eventos que se presentan en el proceso de registrar, editar y borrar los parámetros del sistema
Objetivo	Ingresar los parámetros necesarios para la ejecución del sistema.
Actores	Administrador
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil administrador.
Post condiciones de éxito	• Los parámetros del sistema serán registrados.

Post condiciones de falla	• Los parámetros no serán ingresados. • Se indicará el motivo por el cual no se pudo guardar.
Flujo normal	a) El sistema desplegará un formulario con los controles activados para el ingreso o modificación de los parámetros correspondientes. b) El actor deberá ingresar los datos solicitados. c) El actor salvará la información mediante el botón ***Guardar***. d) El usuario recibe una respuesta con el resultado.
Flujos alternativos	a) En el paso c) del flujo normal: El actor podrá terminar sin guardar mediante un botón ***Cancelar***. b) En el paso d) del flujo normal: Si en el registro de creación o modificación de los parámetros se presentase algún error se realizará una notificación y regresar a la pantalla anterior donde se podrá corregir los errores.

Elaborado por: Paul Jara & Javier Rivera.

Figura 25. Diagrama del caso de uso administración de parámetros



Elaborado por: Paul Jara & Javier Rivera.

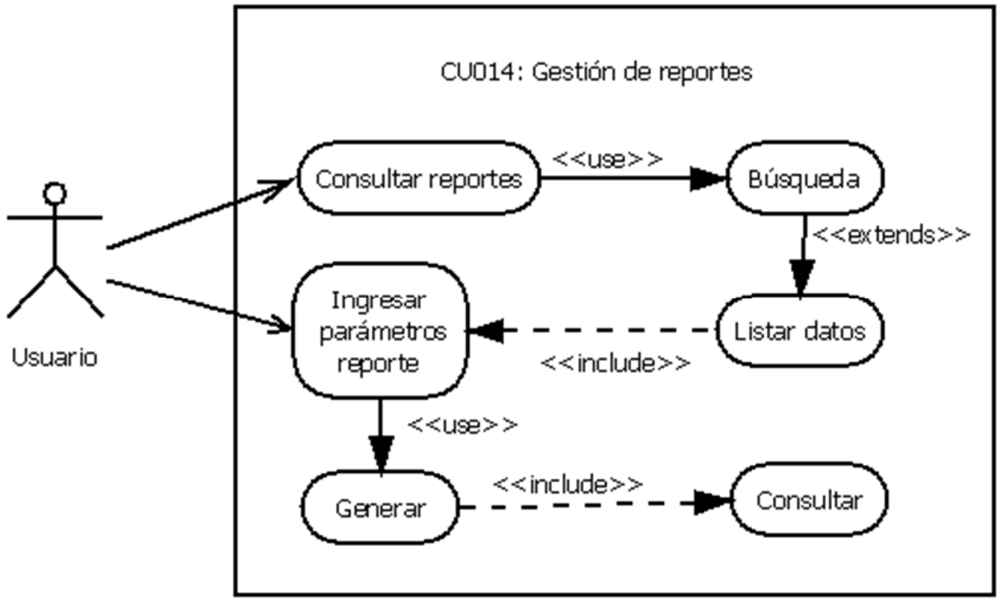
2.4.1.14 Caso de uso gestión reportes.

Tabla 20. Especificación caso de uso gestión reportes

Código	CU014
Nombre	Gestión reportes.
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de presentación de reportes.
Objetivo	Configurar los datos y modelar la información del estado de las diferentes áreas que maneja la empresa, para ser presentada de manera clara y manejable.
Actores	Gerente Administrador Mecánico
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil administrador, mecánico o gerente.
Post condiciones de éxito	• Los usuarios podrán acceder a la información de la empresa presentada en reportes.
Post condiciones de falla	• Los reportes no serán generados. • Se indicará el motivo por el cual no se pudo generar
Flujo normal	a) Se despliega un menú con los diferentes reportes los cuales se mostraran de acuerdo al rol que pertenecen. b) El actor elegirá una opción. c) El actor ingresará los parámetros de entrada del reporte. d) El actor generará el reporte mediante el botón ***Procesar***. e) El actor recibirá una respuesta sobre el proceso.
Flujos alternativos	a) En el paso d) del flujo normal: El actor podrá terminar mediante un botón ***Cancelar***. b) En el paso e) del flujo normal: Si en el proceso se presentase algún error se presentará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 26. Diagrama del caso de uso gestión de reportes



Elaborado por: Paul Jara & Javier Rivera.

2.4.1.15 Caso de uso información de cliente.

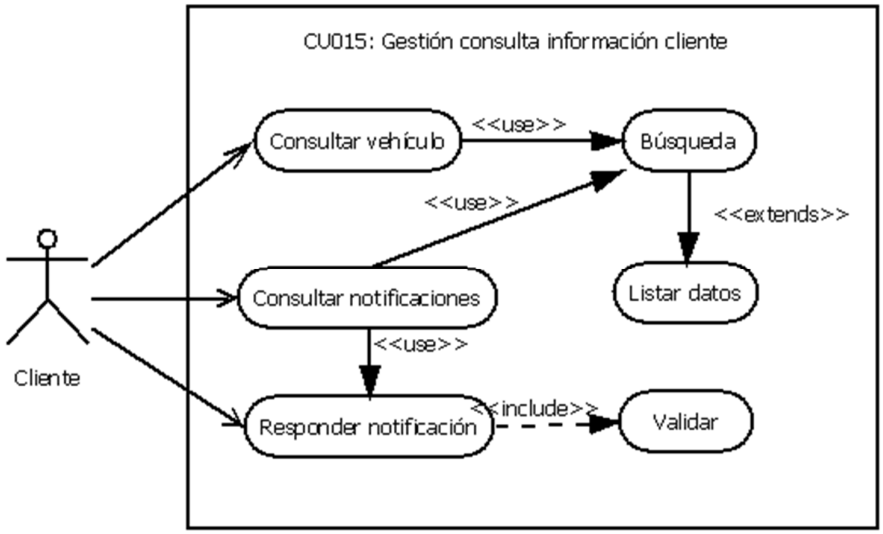
Tabla 21. Especificación caso de uso información del cliente.

Código	CU015
Nombre	Información de cliente
Descripción	En este caso de uso se detallarán los eventos que se presentan en el proceso de presentación de información al cliente.
Objetivo	Consultar la información de importancia del cliente.
Actores	Cliente
Precondiciones	• El actor deberá estar autenticado en el sistema. • El actor deberá tener el perfil cliente.
Post condiciones de éxito	• Los usuarios podrán acceder a la información de importancia presentada.
Post condiciones de falla	• Lo información no será desplegada. • Se indicará el motivo por el cual no se completó el proceso.

Flujo normal	a) Se desplegará un menú con las opciones vehículos y notificaciones. b) El actor elegirá una opción. c) Sí eligiera vehículos, el sistema presentará los datos del vehículo, Sí eligiera notificación se desplegará la información de la notificación.
Flujos alternativos	a) En el paso c) del flujo normal: Al seleccionar un vehículo se deberá mostrar todos los procesos realizados sobre el mismo. b) Si en el proceso se presentase algún se presentará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 27. Diagrama del caso de uso consulta información de cliente



Elaborado por: Paul Jara & Javier Rivera.

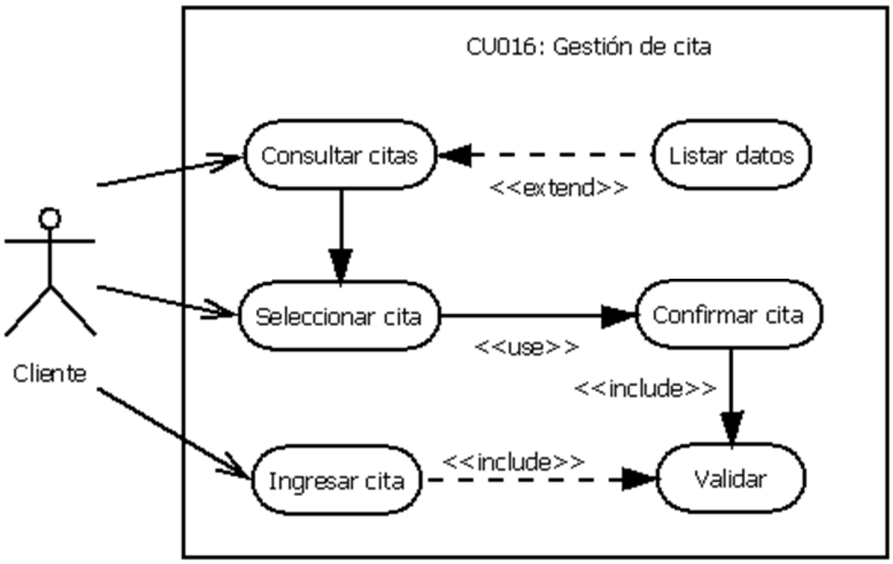
2.4.1.16 Caso de uso gestión de cita.

Tabla 22. Especificación del caso de uso gestión de solicitud de cita.

Código	CU016
Nombre	Gestión de solicitud de cita
Descripción	En este caso de uso se registrarán los eventos que suceden al consultar, editar, confirmar, ingresar solicitudes de Cita.
Objetivo	Permitir al usuario el ingreso, edición, consulta y confirmación de las solicitudes de cita.
Actores	Cliente.
Precondiciones	El actor deberá estar autenticado en el sistema.
Post condiciones de éxito	El usuario podrá obtener información de la solicitud de cita
Post condiciones de falla	Se indicará el motivo por el cual no se pudo realizar el proceso
Flujo normal	a) El flujo principal comenzará cuando el usuario accede a la pestaña citas. b) Se desplegará una pantalla con las solicitudes ingresadas. c) El usuario seleccionará una solicitud de cita. d) EL usuario puede ingresará una nueva solicitud, los datos a ingresar son: descripción, fecha y hora deseada, tipo de revisión. e) El usuario recibirá una respuesta sobre el resultado del proceso.
Flujos alternativos	a) En el paso (b) del flujo normal: b) Si es necesaria la confirmación de la cita, la aplicación habilitará los botones ***Confirmar*** y ***Rechazar***. c) El usuario presionará el botón de su elección. d) En el paso (c) del flujo normal: e) Si en el proceso se presentase algún error se presentará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

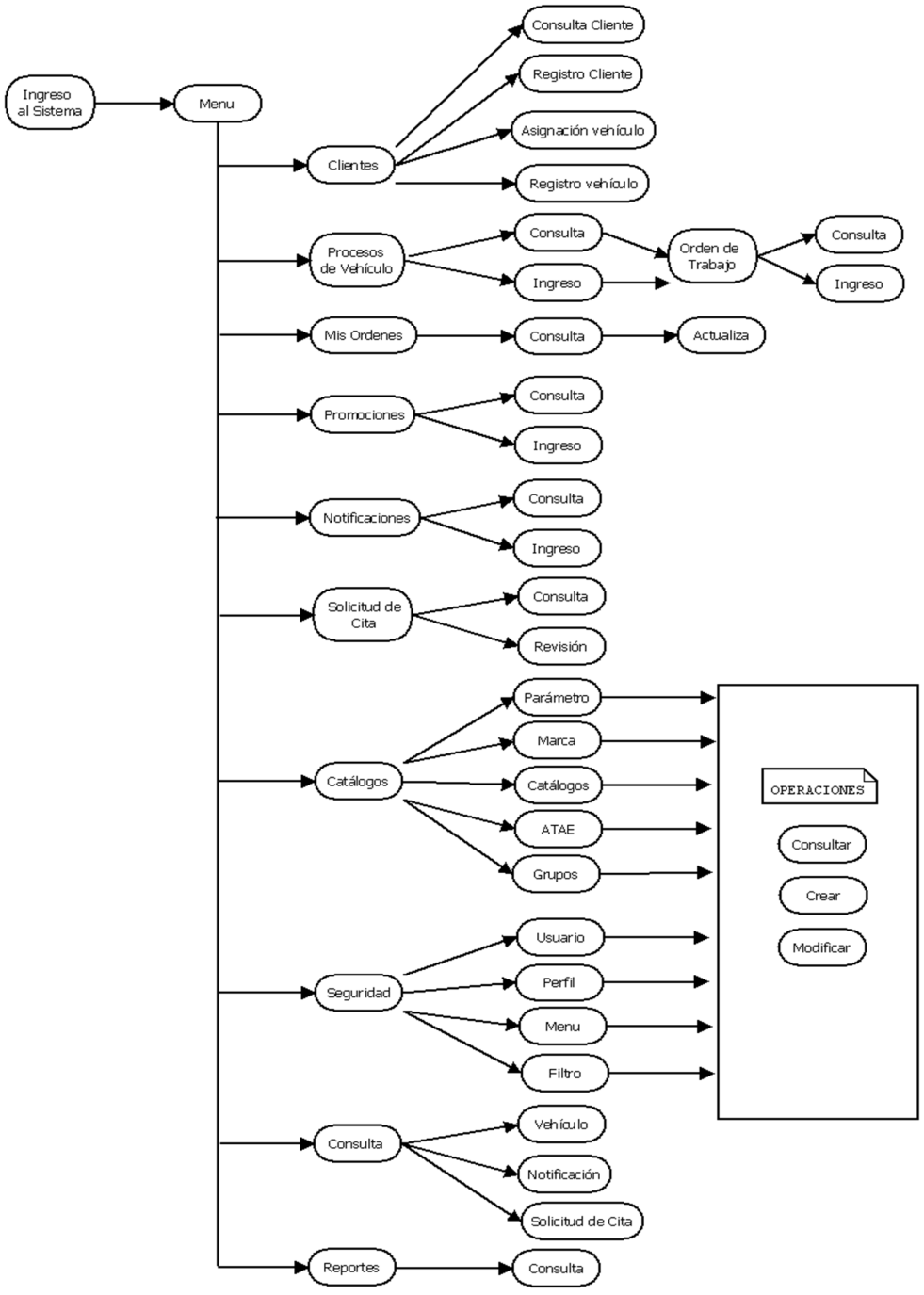
Figura 28. Diagrama del caso de uso gestión de solicitud de cita



Elaborado por: Paul Jara & Javier Rivera.

2.4.2 Diagrama de navegabilidad

Figura 29. Diagrama de navegabilidad



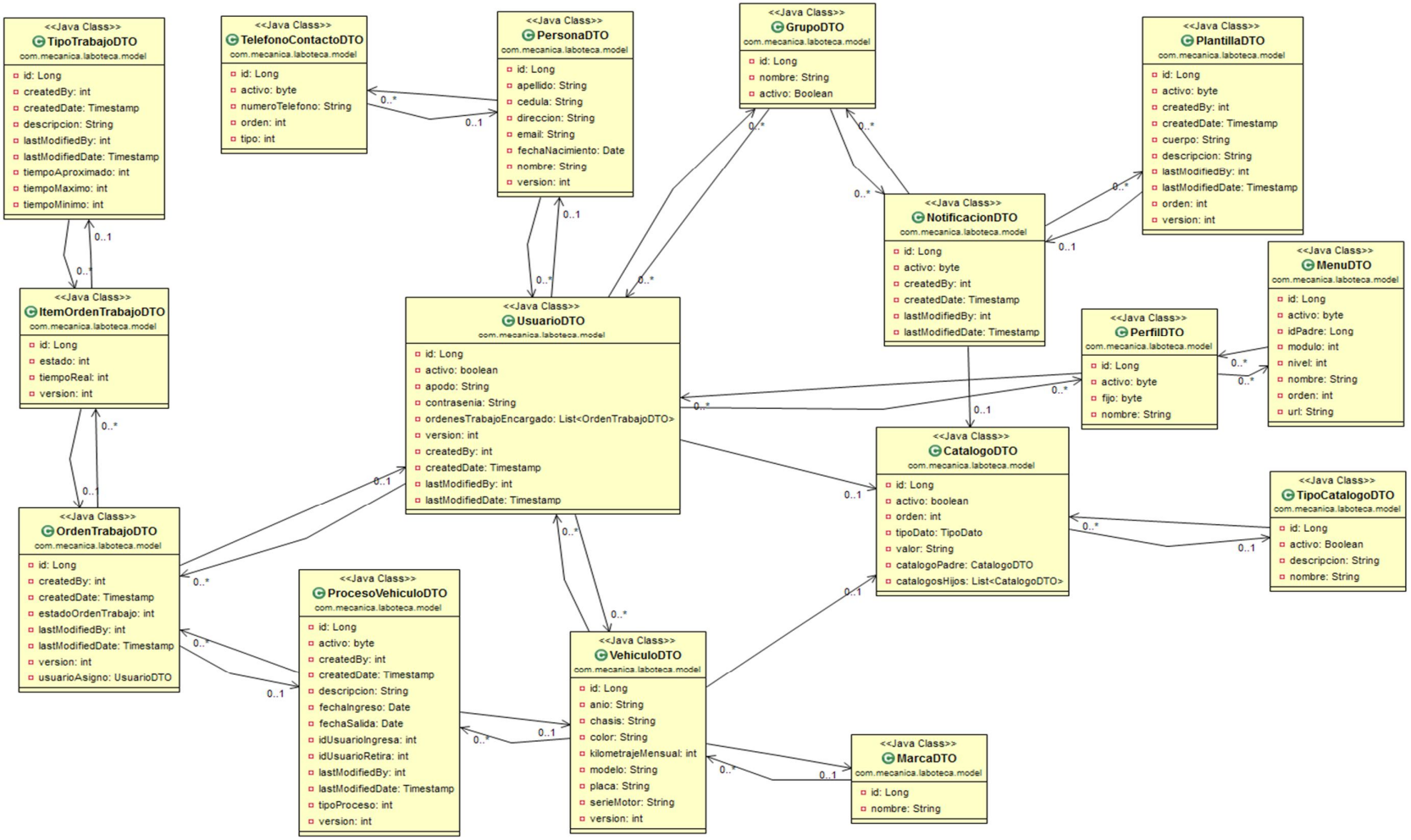
Elaborado por: Paul Jara & Javier Rivera.

2.4.3 Diagrama de clases y entidades

El diagrama de clases representa a las diferentes clases que intervienen en el sistema, así como los tipos de relaciones que mantienen con demás. Para la representación del sistema se empleó dos diagramas:

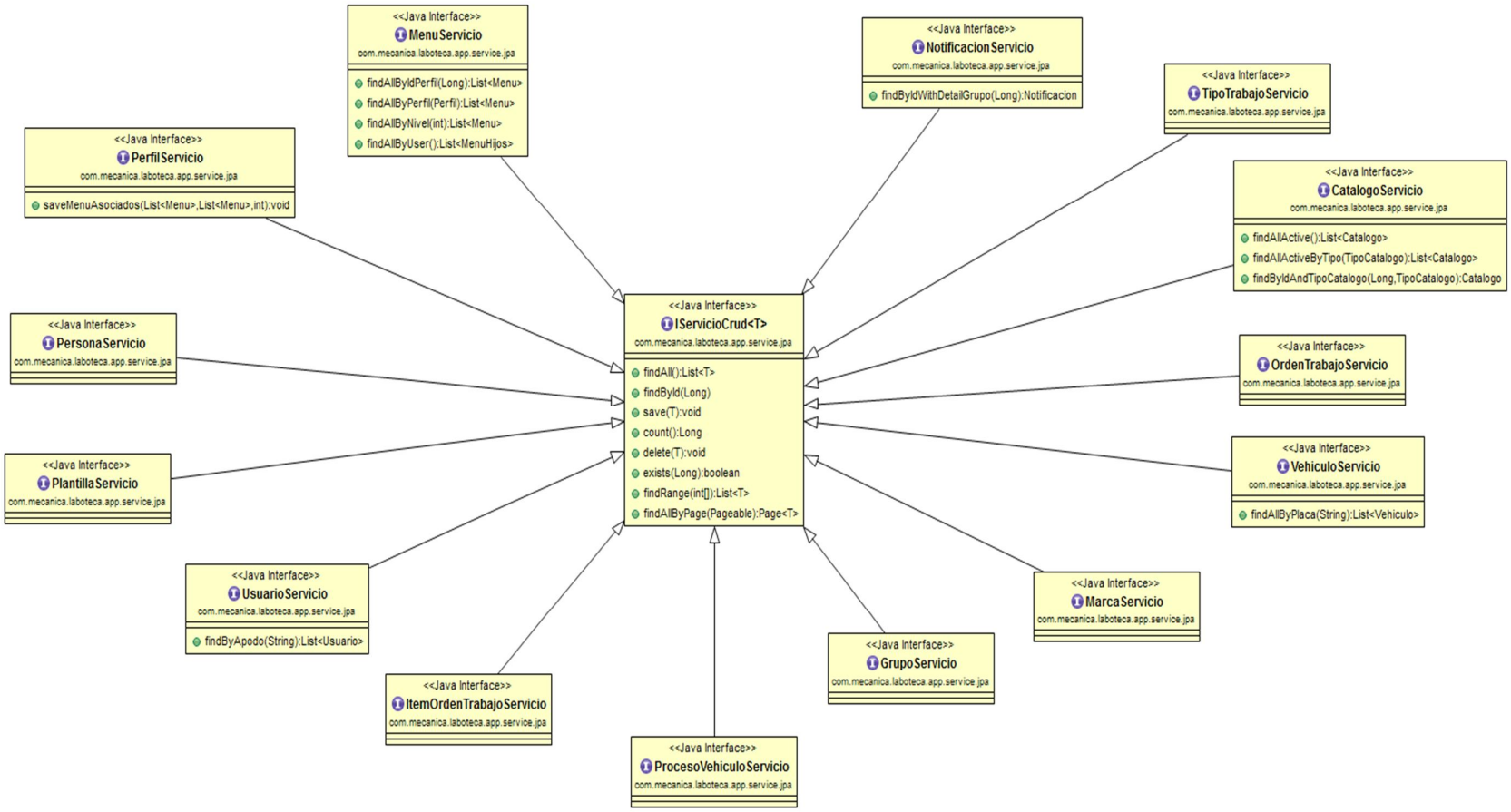
El primero con las entidades correspondientes directamente con la persistencia del JPA, y el segundo con las interfaces de los servicios donde se maneja la lógica del negocio.

Figura 30. Diagrama de entidades



Elaborado por: Paul Jara & Javier Rivera

Figura 31. Diagrama de clases de lógica de negocio



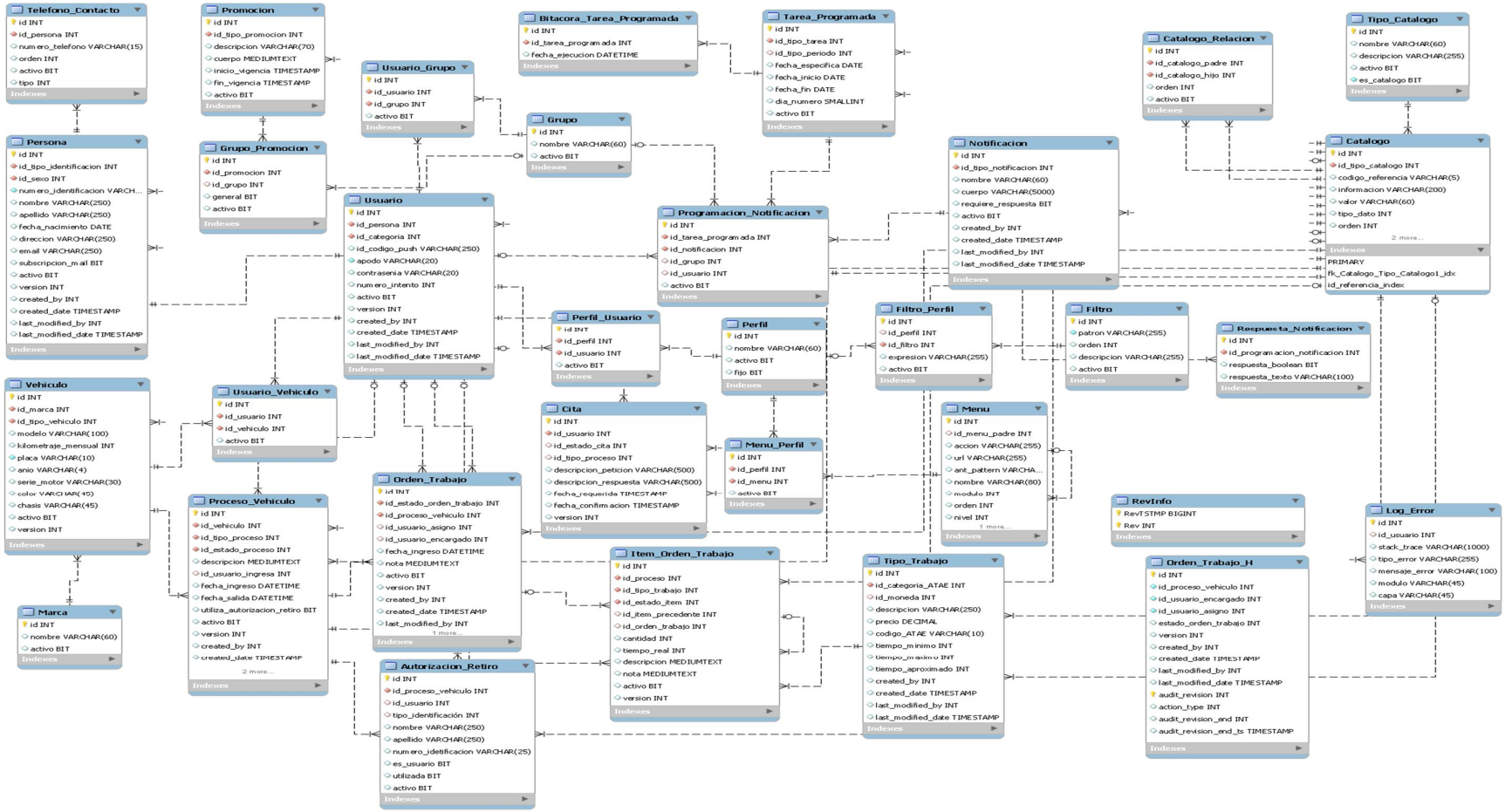
Elaborado por: Paul Jara & Javier Rivera.

2.5 Diseño de datos

2.5.1 Modelo físico.

Para el modelado y generación de la BDD se utilizó la herramienta MySQL WorkBench en su versión 5.2.

Figura 32. Modelo físico



Elaborado por: Paul Jara & Javier Rivera.

CAPÍTULO 3
DISEÑO DEL APLICATIVO MÓVIL

3.1 Planificación del aplicativo

Dentro de las necesidades presentadas por la mecánica Romero Hnos Laboteca, se encuentra, el poder ofrecer la información de la situación de los vehículos de forma inmediata y por medios de fácil acceso para el cliente, contemplando estos puntos y considerando factores como, El gran crecimiento actual del uso de dispositivos móviles, el acceso a Internet de redes de telefonía celular y la robustez del sistema operativo Android, se ofrece la alternativa de implementar una aplicación para Android desde la cual el cliente pueda consultar e informarse sobre los productos y servicios de la mecánica. Para la distribución de la aplicación se colocará un link de descarga en la página web de la mecánica, de esta manera se podrá ofrecer a un mayor número de clientes.

3.2 Requerimientos del aplicativo

El aplicativo móvil debe constar de las siguientes funcionalidades.

- Autenticación en la aplicación.
- Ofrecer información al cliente del estado de los servicios realizados sobre su(s) vehículo(s).
- Ofrecer información al cliente sobre promociones y productos.
- Entregar notificaciones inmediatas sobre asuntos de interés del cliente y permitirle su interacción en caso de ser requerida.
- El sistema debe permitir consultar o ingresar una solicitud de cita para atención en la mecánica, esta será respondida por el personal encargado de esta actividad.

Solución

- **Módulo de información para dispositivos móviles**: permitir que los clientes soliciten citas, realicen seguimiento de sus vehículos y revisen el historial de cada uno, a través de una aplicación para dispositivos móviles que cuenten con sistema operativo Android.

3.3 Diseño y arquitectura

La aplicación está diseñada para ejecutarse sobre el sistema operativo Android en su versión 4.0 (Ice Cream Sandwich) API 14, una versión estable con características que se enfocan principalmente en ejecutar tareas simultaneas y mejorar la experiencia de usuario, con acciones comunes más visibles, además permite navegar con gestos sencillos e intuitivos.

Utilizar los recursos eficientemente a la hora de desarrollar aplicaciones móviles es una piedra angular que no se puede obviar, especialmente en el consumo de datos de Internet, debe ser optimizado al máximo posible con la finalidad de ahorrar batería y aprovechar las tarifas de datos existentes. En este sentido JSON, gracias a su estructura, es un ahorro considerable respecto a XML y representa la mejor opción para intercambio de datos entre la aplicación y el servidor web.

Para enviar notificaciones de manera inmediata al cliente, se optó por utilizar mensajes Push, la ventaja de este tipo de mensajes es que son administrados directamente por el sistema operativo y no requiere que la aplicación se encuentre desplegada al momento de recibirlos.

Para la implementación de mensajes Push, se realizará a través del servicio GCM (Google Cloud Messaging for Android), un servicio gratuito proporcionado por Google que permite enviar datos de un servidor a un dispositivo con Android y/o recibir mensajes de los dispositivos en la misma conexión. Este servicio gestiona todos los aspectos de gestión de colas de mensajes y la entrega a la aplicación

Android de destino que se ejecuta en el dispositivo de destino. (developer.android.com, 2012)

El desarrollo de la aplicación está encaminado a ofrecer la mayor compatibilidad posible con las versiones anteriores de Android, con el objetivo de llegar a más clientes que utilicen dispositivos antiguos.

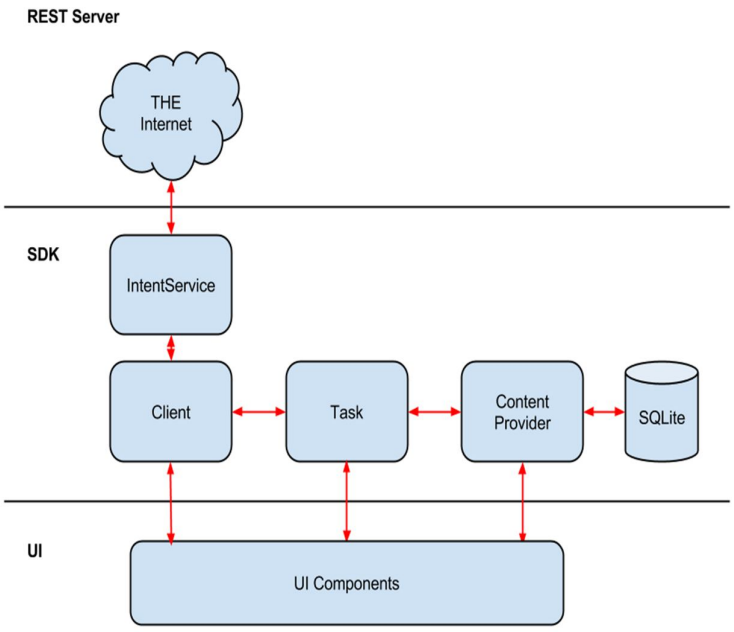
La arquitectura considerada para la aplicación móvil está basada en tres capas.

REST Server se encuentra del lado del servidor, ejecuta las peticiones enviadas desde la aplicación móvil y maneja la persistencia en la base de datos del sistema web.

SDK esta capa es la que se encarga de la entrada y salida de datos hacia el servidor, maneja los intents (invocadores de componentes), además realiza la persistencia en la base de datos de Android. (SQLite).

UI Components en esta capa esta todo aquello encargado de gestionar la interfaz: objetos propios de Android como activities, fragments, views, adapters, etc; Esta encaminada a la presentación de la aplicación móvil.

Figura 33. Arquitectura de la aplicación Android



Fuente: sergiandreplace.com

3.3.1 Diagramas UML.

3.3.1.1 Diagramas de casos de uso.

Los roles que interactúan con el aplicativo móvil son:

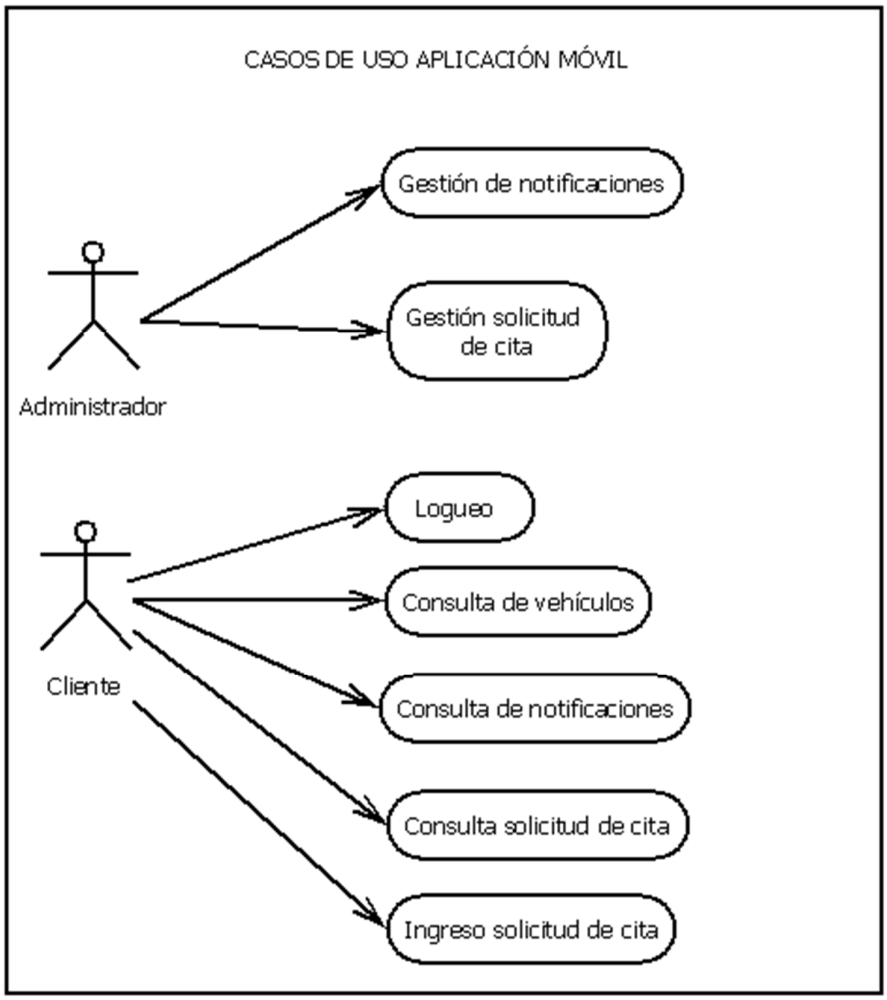
Tabla 23. Actores del aplicativo móvil.

Núm.	Actor	Descripción
1	Administrador	Es la persona encargada de la gestión de datos relacionados al sistema, como la configuración de parámetros y asignación de permisos.
2	Cliente	Interactúa con las diferentes opciones que le ofrece la aplicación.

Elaborado por: Paul Jara & Javier Rivera.

La aplicación móvil contiene los siguientes casos de uso:

Figura 34. Casos de uso de la aplicación móvil



Elaborado por: Paul Jara & Javier Rivera.

Los casos de uso para el administrador se encuentran detallados dentro del sistema web.

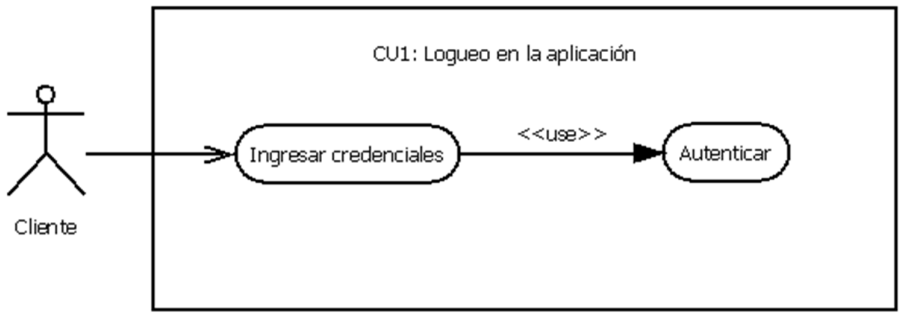
3.3.1.2 Caso de uso autenticación en la aplicación móvil.

Tabla 24. Especificación del caso de autenticación en la aplicación móvil

Código	CUM001
Nombre	Autenticación en la aplicación móvil.
Descripción	En este caso de uso se registrarán los eventos que sucederán al efectuar el proceso de autenticación desde la aplicación móvil.
Objetivo	Registrar y validar el ingreso de un usuario el aplicativo móvil.
Actores	Cliente
Precondiciones	• El usuario deberá estar registrado en el sistema. • El usuario deberá contar con la aplicación instalada en su dispositivo móvil.
Post condiciones de éxito	• El usuario podrá interactuar con las diferentes funcionalidades que ofrece la aplicación.
Post condiciones de falla	• Se indicará el motivo por el cuál no fue permitido el acceso a la aplicación.
Flujo normal	El flujo principal comenzará cuando el usuario accede a la aplicación. a) Se desplegará una pantalla con los campos habilitados para el ingreso de las credenciales necesarias. b) El usuario deberá ingresar la siguiente información: nombre de usuario y contraseña. c) El usuario presionará el botón ***Ingresar*** para ejecutar el proceso de autenticación. d) El usuario recibirá una respuesta sobre el proceso. e) La aplicación registrará el identificador para GCM del dispositivo en el sistema. f) La aplicación presentará una pantalla con tabs de las diferentes opciones de la aplicación.
Flujos alternativos	a) En el paso (d) del flujo normal: Si en el proceso se suscitara algún error la aplicación mostrará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 35. Diagrama del caso de uso autenticación en la aplicación móvil



Elaborado por: Paul Jara & Javier Rivera.

3.3.1.3 Caso de uso consulta de vehículos.

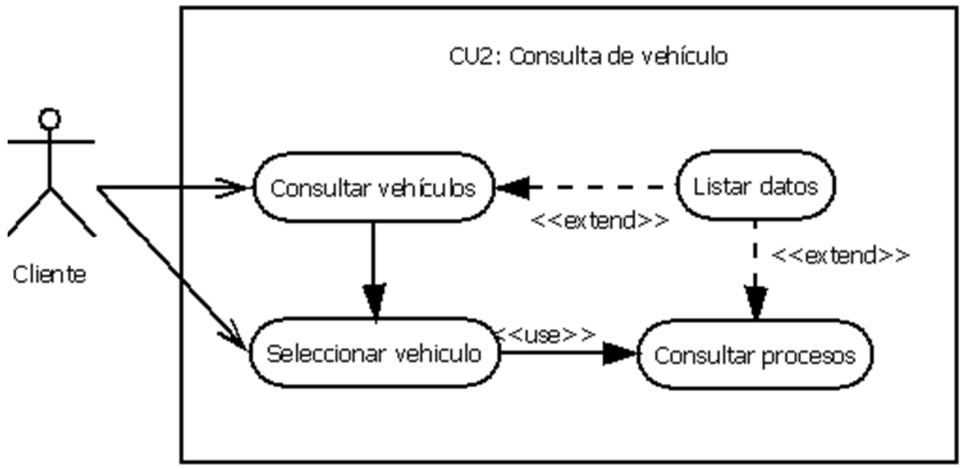
Tabla 25. Especificación del caso de uso consulta de vehículos

Código	CUM002
Nombre	Consulta de vehículos.
Descripción	En este caso de uso se registrarán los eventos que sucederán al efectuar una consulta sobre los vehículos registrados.
Objetivo	Presentar al usuario de manera ordenada y amigable la información relevante de los vehículos registrados y los trabajos realizados sobre estos.
Actores	Cliente
Precondiciones	• El actor deberá estar autenticado en el sistema.
Post condiciones de éxito	• El usuario podrá acceder a la información solicitada.
Post condiciones de falla	• Se indicará el motivo por el cual la información no puede ser presentada.
Flujo normal	El flujo principal comenzará cuando el usuario accede a la aplicación. a) Se desplegará una pantalla con la lista de vehículos registrados. b) El usuario seleccionará un vehículo. c) El usuario recibirá una respuesta con los trabajos realizados sobre el vehículo. d) El usuario seleccionará un trabajo realizado.

	e) El usuario recibirá una respuesta con el detalle del trabajo realizado.
Flujos alternativos	a) En el paso (c) del flujo normal: Si en el proceso se suscitara algún error la aplicación mostrará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 36. Diagrama del caso de uso consulta de vehículos



Elaborado por: Paul Jara & Javier Rivera.

3.3.1.4 Caso de uso consulta de notificaciones.

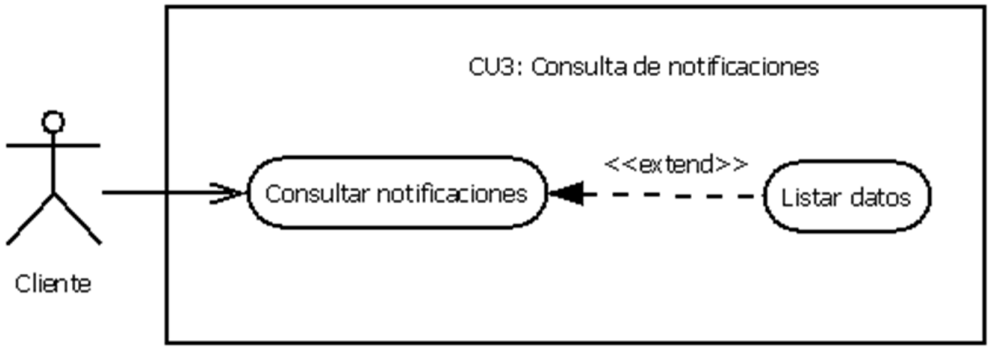
Tabla 26. Especificación del caso de uso consulta de vehículo.

Código	CUM003
Nombre	Consulta de notificaciones
Descripción	En este caso de uso se registrarán los eventos que sucederán al efectuar una consulta sobre las notificaciones que ofrece la aplicación móvil.
Objetivo	Presentar al usuario de manera ordenada y amigable las notificaciones enviadas por la mecánica.
Actores	Cliente
Precondiciones	• El actor deberá estar autenticado en el sistema

Post condiciones de éxito	• El usuario podrá acceder a la información solicitada
Post condiciones de falla	• Se indicará el motivo por el cual la información no puede ser presentada
Flujo normal	El flujo principal comienza cuando el usuario accede a la aplicación. a) Se desplegará una pantalla con la lista de notificaciones en caso de existir. b) El usuario seleccionará una notificación. c) El usuario recibirá una respuesta con el detalle de la notificación.
Flujos alternativos	a) En el paso(a) se desplegará una pestaña adicional con las promociones disponibles. b) En el paso (c) del flujo normal: i. Si la notificación requiere ser respondida por el usuario el sistema deberá habilitar la caja de texto para el ingreso de la respuesta, al igual que un botón que le permitirá enviar dicha respuesta. ii. Si en el proceso se suscitara algún error la aplicación mostrará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 37. Diagrama del caso de uso consulta de notificaciones



Elaborado por: Paul Jara & Javier Rivera.

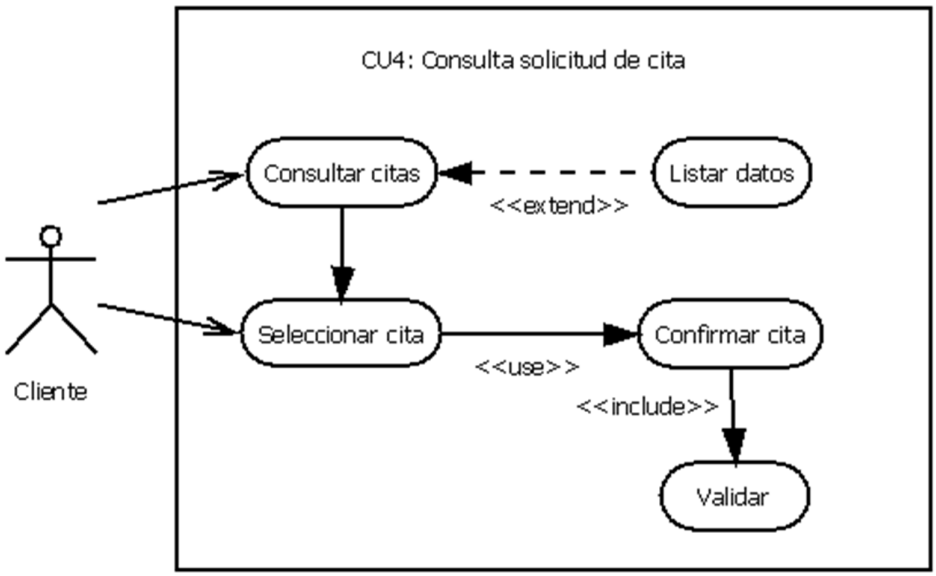
3.3.1.5 Caso de uso consulta de solicitud de cita.

Tabla 27. Especificación del caso de uso consulta de solicitud de cita

Código	CUM004
Nombre	Consulta de solicitud de cita
Descripción	En este caso de uso se registrarán los eventos que sucederán al consultar las solicitudes de cita ingresadas
Objetivo	Permitir al usuario consultar los detalles de una solicitud de cita ingresada en el sistema
Actores	Cliente
Precondiciones	• El actor deberá estar autenticado en el sistema
Post condiciones de éxito	• El usuario podrá obtener información de la solicitud de cita
Post condiciones de falla	• Se indicará el motivo por el cual no se pudo realizar el proceso
Flujo normal	El flujo principal comenzará cuando el usuario accede a la pestaña citas. a) Se desplegará una pantalla con las últimas solicitudes ingresadas. b) El usuario seleccionará una solicitud de cita. c) El usuario recibirá una respuesta sobre el resultado del proceso.
Flujos alternativos	a) En el paso (b) del flujo normal: Si es necesaria la confirmación de la cita, la aplicación habilitará los botones ***Confirmar*** y ***Rechazar***. El usuario presionará el botón de su elección. b) En el paso (c) del flujo normal: Si en el proceso se suscitara algún error la aplicación mostrará una notificación del problema.

Elaborado por: Paul Jara & Javier Rivera.

Figura 38. Diagrama del caso de uso consulta de solicitud de cita



Elaborado por: Paul Jara & Javier Rivera.

3.3.1.6 Caso de uso ingreso de solicitud de cita.

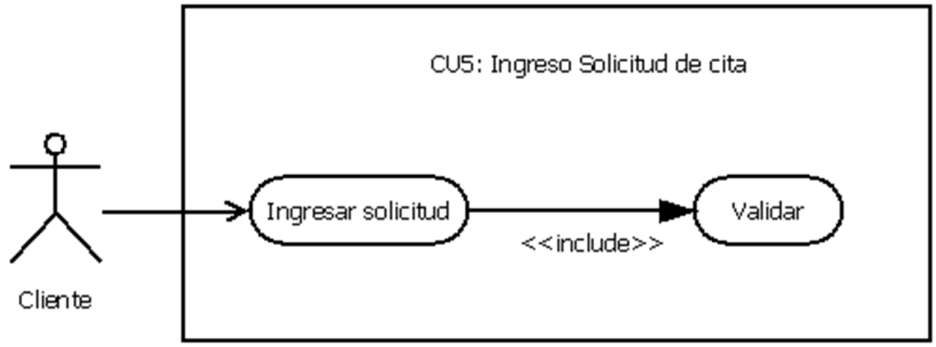
Tabla 28. Especificación del caso de uso ingreso de solicitud de cita.

Código	CUM005
Nombre	Ingreso de solicitud de cita.
Descripción	En este caso de uso se registrarán los eventos que sucederán al ingresar una solicitud para una cita en la mecánica.
Objetivo	Permitir al usuario ingresar una solicitud para ser atendido en la mecánica en una fecha y horario de su conveniencia.
Actores	Cliente.
Precondiciones	• El actor deberá estar autenticado en el sistema.
Post condiciones de éxito	• El usuario podrá disponer de su cita, si es aceptada por la mecánica.
Post condiciones de falla	• Se indicará el motivo por el cual no se pudo realizar el proceso.

Flujo normal	El flujo principal comenzara cuando el usuario presiona el botón de ingreso solicitud de cita. a) Se desplegará una pantalla con los campos necesarios para el registro. b) El usuario deberá ingresar: fecha, hora, tipo de trabajo a realizar, descripción. c) El usuario recibirá una respuesta sobre el resultado del proceso.
Flujos alternativos	a) En el paso (c) del flujo normal: Si en el proceso se suscitara algún error la aplicación mostrará una notificación del problema.

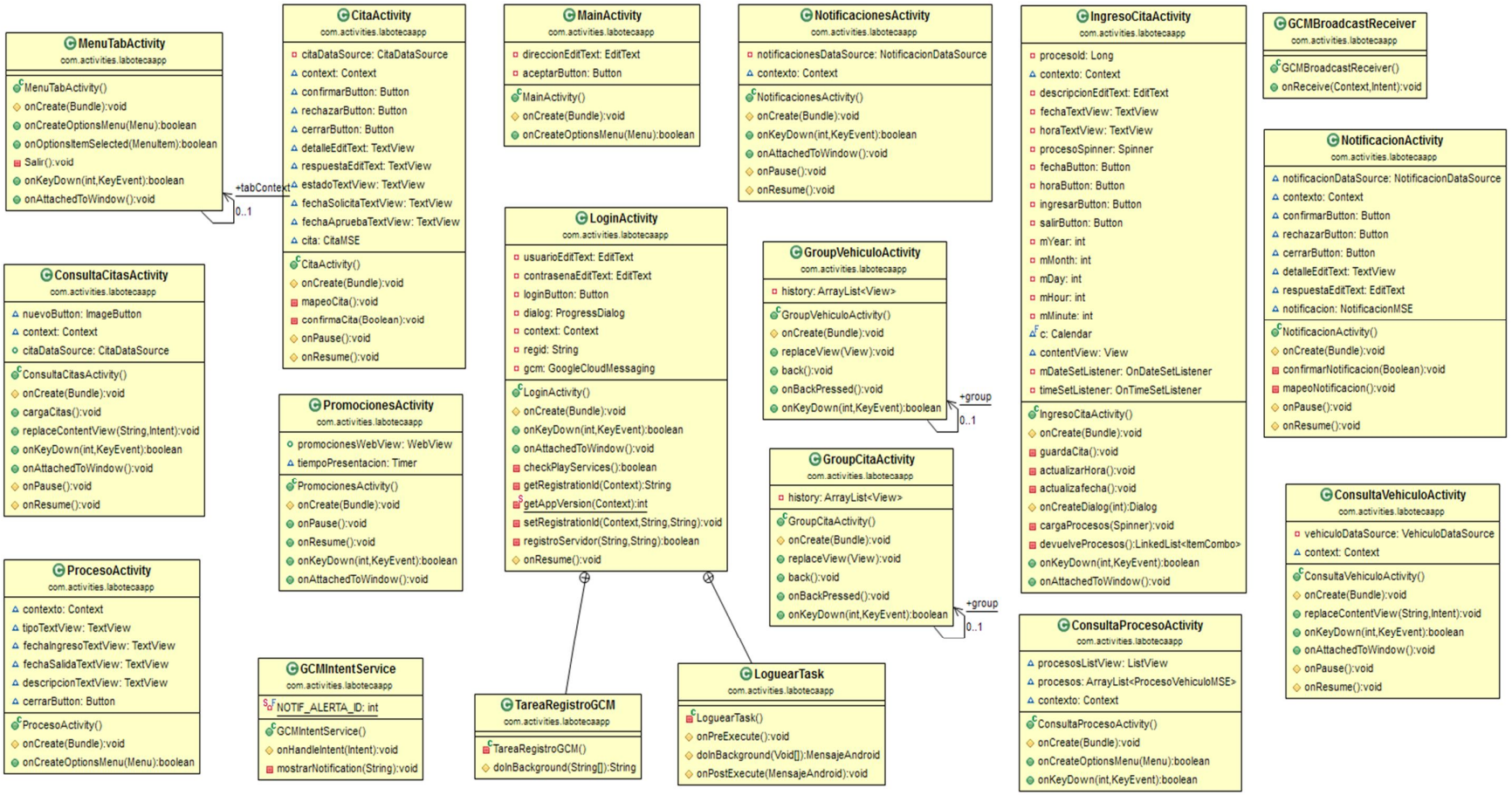
Elaborado por: Paul Jara & Javier Rivera.

Figura 39. Diagrama del caso de uso ingreso de solicitud de cita



Elaborado por: Paul Jara & Javier Rivera.

Figura 40. Diagrama de clases de la aplicación móvil



Elaborado por: Paul Jara & Javier Rivera.

CAPÍTULO 4
DESARROLLO E INTEGRACIÓN

4.1 Desarrollo de la aplicación.

4.1.1 Herramientas utilizadas.

El sistema web está desarrollado con las siguientes aplicaciones:

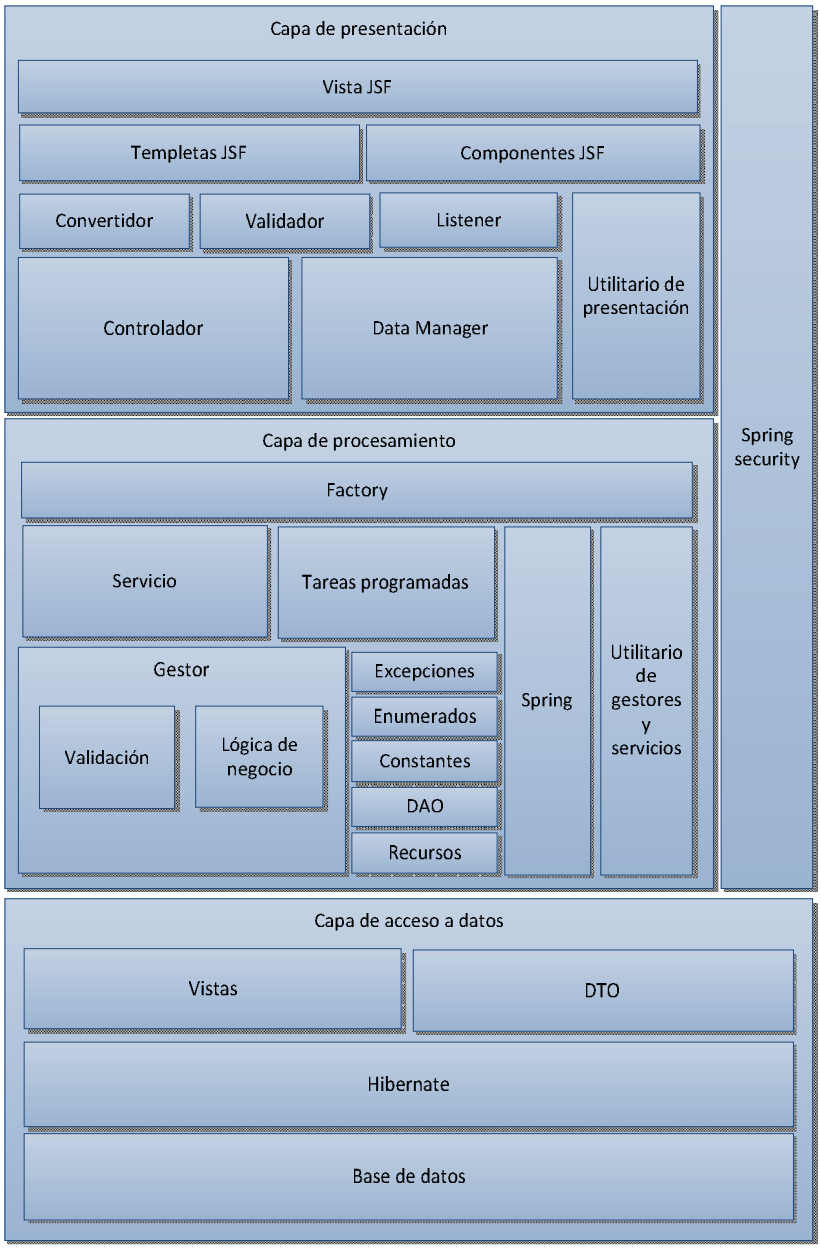
- Eclipse Spring Tool Suite 3.4 como IDE para desarrollo.
- MySQL 5.1 como motor de base de datos.
- JBoss Community 7.1 como servidor de aplicaciones.
- Richfaces 4.3.4 como librería de componentes.
- Primefaces 4.0 como librería de componentes.
- JSF 2.0 como framework base para el desarrollo de la aplicación.

La aplicación móvil está desarrollada para funcionar sobre dispositivos móviles que tengan el sistema operativo Android versión 4.0.

4.1.2 Arquitectura sistema web.

El sistema web está basada en una arquitectura N capas agrupadas en tres grupos como se describe en el capítulo 2.

Figura 41. Arquitectura final del sistema web



Elaborado por: Paul Jara & Javier Rivera.

4.1.2.1 Capa de presentación.

La capa de presentación recoge la información de los usuarios, presenta datos, controla la navegación entre páginas, valida los datos ingresados y mantiene el estado de la sesión de la aplicación web, para la capa de presentación se utiliza JSF y MVC.

Controlador.

Los controladores en JSF son ManagedBeans donde se ubican todos los procesos principales para una pantalla, el alcance de estos beans puede ser Request o View. El alcance Request solo mantiene el estado del bean durante el tiempo que dura una petición y este tiempo es corto, View mantiene el estado del bean mientras se mantenga en la misma vista o página, esto disminuye la cantidad de variables en sesión, haciendo que las sesiones ocupen menos espacio en el servidor.

Figura 42. Controlador de órdenes de trabajo

```
@ManagedBean
@ViewScoped
public class OrdenTrabajoController extends CommonController{
  public static final String PAGE_CREAR_ORDEN_TRABAJO =
"/pages/ordenTrabajo/admin/crearOrdenTrabajo";
  @ManagedProperty(value="#{ordenTrabajoDataManager}")
  private OrdenTrabajoDataManager ordenTrabajoDataManager;

  public String crearOrdenTrabajoAction(){
    return PAGE_CREAR_ORDEN_TRABAJO + "?faces-redirect=true";
  }
```

Elaborado por: Paul Jara & Javier Rivera.

En el código de la figura 42 se utiliza la anotación @ManagedBean para que el servidor de aplicaciones reconozca a esta clase como un bean administrado de JSF, este puede contener la navegación de la aplicación, que de otra forma debería ser colocado en el archivo de configuración faces-config.xml

También se utiliza la anotación @ViewScoped para definir que el alcance del bean sea de vista es decir que dure mientras se encuentre en la página, cuando se dirija hacia otra vista o página de la aplicación web este bean será destruido.

La anotacion @ManagedProperty recibe el parámetro (value) en el cual se asigna el nombre de otro bean de JSF llamado ordenTrabajoDataManager, para poder acceder a los valores almacenados en sesión que se encuentran en el datamanager.

Dentro del controlador se especifica la navegación de la aplicación a modo de comando, es decir desde la vista se ejecuta una acción que redirigirá la página del navegador hacia otra sección de la aplicación, en este caso la dirección de la página se encuentra contenida en la variable PAGE_CREAR_ORDEN_TRABAJO, la acción entrega la dirección de la página en la aplicación web.

Figura 43. PostContruct para el controlador OrdenTrabajoController

```
//Beans
  private CrearOrdenTrabajo crearOrdenTrabajo;
  @PostConstruct
  private void postContruct(){
    log.info(FacesContext.getCurrentInstance().getViewRoot().getViewId());
    if
(FacesContext.getCurrentInstance().getViewRoot().getViewId().equals(PAGE_CREAR_ORDEN_TRA
BAJO + ".xhtml")) {
      if(JsfUtil.getRequestParameter("edit") != null){
        crearOrdenTrabajo = new CrearOrdenTrabajo(ordenTrabajoDataManager.getOrdenTrabajo(),
getUsuarioSession().getId());
      } else if (procesoVehiculoDataManager.getProcesoVehiculo() != null){
        crearOrdenTrabajo = new
CrearOrdenTrabajo(procesoVehiculoDataManager.getProcesoVehiculo().getId(),
getUsuarioSession().getId());
      }
```

Elaborado por: Paul Jara & Javier Rivera.

En la figura 43 muestra el método postConstruct que se encarga de inicializar backing beans que contienen la lógica de presentación para cada página a la que se dirija la aplicación web, para realizar esta operación se utiliza el utilitario JsfUtil para obtener los parámetros enviados al controlador y el FacesContext que brinda acceso

a la información de la página, este método tiene la anotación @PostConstruct que indica a JSF que una vez instanciado el bean se ejecute este método.

Datamanager.

Los datamanagers o manejadores de datos son ManagedBeans que tienen un alcance de sesión, es decir el estado de esté bean se mantiene mientras el usuario no cierre el navegador o salga de la aplicación web, cerrando la sesión. En estos beans solo se almacena lo que se necesite, no se guardan objetos pesados ni procedimientos.

Vista

Facelets es el sistema por defecto para crear vistas basadas en árboles de componentes JSF, además es el sistema de plantillas estándar en la versión 1.2 y 2 de JSF y forma parte de la propia especificación.

La implementación de la capa de presentación implica crear las páginas JSF y definir las ubicaciones de estas en constantes ubicadas en los controladores o en el archivo faces-config.xml para a través de estas definir la navegación entre páginas, también implica acceder a los beans de presentación desde las páginas, mediante el uso de lenguaje de expresiones que facilita la presentación de los datos y el acceso a la lógica de presentación, como eventos, acciones, procedimientos y validaciones.

Figura 44. Uso de lenguaje de expresiones una página JSF

```
<co:inputText label="Nombre" sizeLabel="80"
value="#{ordenTrabajoController.crearOrdenTrabajo.ordenTrabajo.usuarioEncargado.apellidosYNombres}" readonly="true"
rendered="#{ordenTrabajoController.crearOrdenTrabajo.ordenTrabajo.usuarioEncargado.id != null}" />
<a4j:commandLink value="Buscar"
  oncomplete="#{rich:component('ppBuscarUsuario')}.show();"
  render="pgBusquedaUsuario" />
```

Elaborado por: Paul Jara & Javier Rivera.

En la figura 44 se muestra el uso de lenguaje de expresiones para acceder a la propiedad “apellidosYNombres” que contiene los nombres y apellidos del usuario, este dato es colocado en un componente personalizado que tiene una caja de texto y una etiqueta.

Utilitarios.

Los utilitarios contienen métodos estáticos que contienen funcionalidades comunes que son utilizadas en varias partes del sistema, se encuentran ubicados en el paquete "com.mecanica.laboteca.web.util".

En la figura 45 se muestra un método de la clase JsfUtil, desde la cual se maneja el contexto de la aplicación, los mensajes de alerta y excepciones generados en los procesos, el método *addErrorMessage* despliega los mensajes de error en las páginas JSF del sistema web.

Figura 45. Método de la clase JsfUtil para el envío de mensajes de error

```
public static void addErrorMessage(String msg) {
    FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, msg);
    // captura contexto
    FacesContext.getCurrentInstance().addMessage(null, facesMsg);
  // envía el mensaje a la vista
 }
```

Elaborado por: Paul Jara & Javier Rivera.

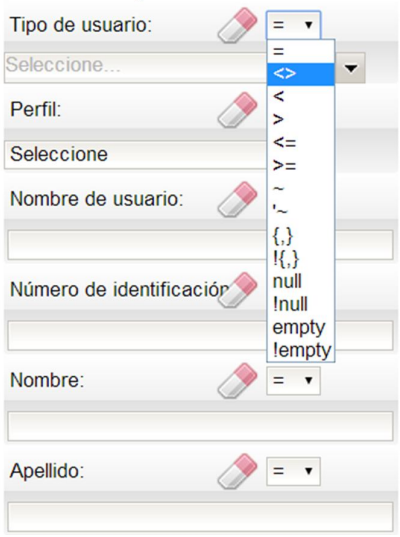
Componentes.

Hay varios componentes JSF que se van a reutilizar en todo el sistema, y se encuentran bajo el directorio “/WebContent/resources/components”, es en esta carpeta donde la especificación indica que JSF debe buscar los componentes del sistema, y permite utilizar estos componentes en varias páginas mediante el uso de etiquetas.

Para realizar búsquedas existen componentes genéricos que son ocupados para filtrar datos y presentarlos en las páginas del sistema web, esto permite mantener una interface de usuario estándar y facilita la implementación de los componentes.

En la figura 46 se muestra un grupo de filtros de búsqueda implementados para la consulta de usuarios registrados en el sistema, el componente cuenta con un botón para resetear el filtro, con un lista de opciones operaciones lógicas para la búsqueda y con un campo donde el usuario ingresa los datos para el componente, si se resetea el filtro, este regresa a su estado inicial nulo y no afectará al resultado de consulta que se realice contra la base de datos.

Figura 46. Diseño de filtros de búsqueda



Elaborado por: Paul Jara & Javier Rivera.

En la figura 47 se muestra el método encargado de inicializar un componente con el que se va a realizar la consulta para vehículos, el método utiliza entidades de Hibernate y permite formar una consulta al estilo de Criteria Query, mediante un case, se revisa si el enumerado coincide con la opción de la estructura para proceder con su inicialización, en este caso se indica que el filtro afectara a la propiedad "usuarioVehiculos.vehiculo.placa". La librería generic-dao es la encargada de realizar las uniones (join) de las tablas en la consulta SQL basándose en Hibernate.

Figura 47. Inicialización de filtros

```
private void initFiltro(FiltrosUsuarioComponentEnum filtro) {
    switch (filtro) {
    case filtroPlaca:
      filtroPlaca = new FiltroBusquedaComponent<String>(String.class,
          "usuarioVehiculos.vehiculo.placa");
      break;
    //Código omitido
    }
  }
```

Elaborado por: Paul Jara & Javier Rivera.

Estos datos se envían al servicio que se encarga de interpretarlos y entrega el resultado de la consulta de manera que no sea necesario crear la misma interfaz para cada campo que se desea filtrar en la consulta, permitiendo de esta manera optimizar los recursos y reutilizar código en varias capas del sistema web, se debe recolectar la los datos ingresados en los componentes. Como se muestra en la figura 48 se verifica si el valor del componente es distinto de nulo para agregarlo a una colección que contiene la información de los componentes mostrados en la página web.

Figura 48. Recolección datos de los filtros

```
@SuppressWarnings("rawtypes")
 public void buscarUsuarios() {
  Collection<FiltroComponent> filtros = new ArrayList<FiltroComponent>(0);
  if (filtroPlaca != null
      && StringUtils.isNotEmpty(filtroPlaca.getValue())) {
    filtros.add(filtroPlaca);
  }
  this.usuariosEncontrados = MecanicaFactory.getUsuarioService()
      .obtenerUsuariosPersona(filtros);
 }
```

Elaborado por: Paul Jara & Javier Rivera.

Validadores, convertidores y escuchas.

En capa de presentación se valida que el usuario ingrese los datos en la pantalla y que estos sean del tipo correcto (entero, cadena, cédula, etc.) estas validaciones extienden de la clase javax.faces.validator.Validator.

Los **convertidores** son los encargados de convertir los datos de un tipo en Java a otro tipo para presentarlo en la vista, o viceversa y extienden de la clase javax.faces.convert.Converter.

Los **escuchadores** o listeners son los encargados de realizar algún procedimiento cuando se realiza alguna acción en la aplicación web, estas clases extienden de la clase javax.faces.event.PhaseListener.

Estas clases están ubicadas bajo el paquete "com.mecanica.laboteca.web.common", cada una tiene un subpaquete respectivamente.

4.1.2.2 Capa de procesamiento.

En esta capa se encuentra la lógica de negocio expuesta en servicios de Spring y varios utilitarios, encargados de proveer las funcionalidades para realizar operaciones con los objetivos del sistema.

Servicios

La lógica de negocio se encuentra expuesta a manera de servicios y para ser utilizada desde la capa de presentación es necesario utilizar la clase MecanicaFactory que se detalla en la sección 4.2.2, estos servicios tienen la anotación @Service y la anotación @Transactional en las clases que los implementan, Spring va a reconocer las clases que tengan estas anotaciones como beans de servicio y se encarga de crear una unidad de persistencia por cada llamada a uno de los métodos que se encuentren dentro estas a menos que se indique lo contrario.

El código que se presenta en la figura 49, muestra la clase OrdenTrabajoService que tiene en la parte superior las anotaciones antes indicadas. También está la implementación del método obtenerOrdenTrabajoDetallesPorId este método solo puede realizar operaciones de lectura contra la base de datos ya que tiene la anotación @Transactional con el parámetro readOnly con valor igual a true, esto quiere decir que las operaciones realizadas por este método solo pueden ser de lectura, esto resulta en una transacción más rápida de ejecutar y liviana para el servidor.

Figura 49. Estructura de un servicio

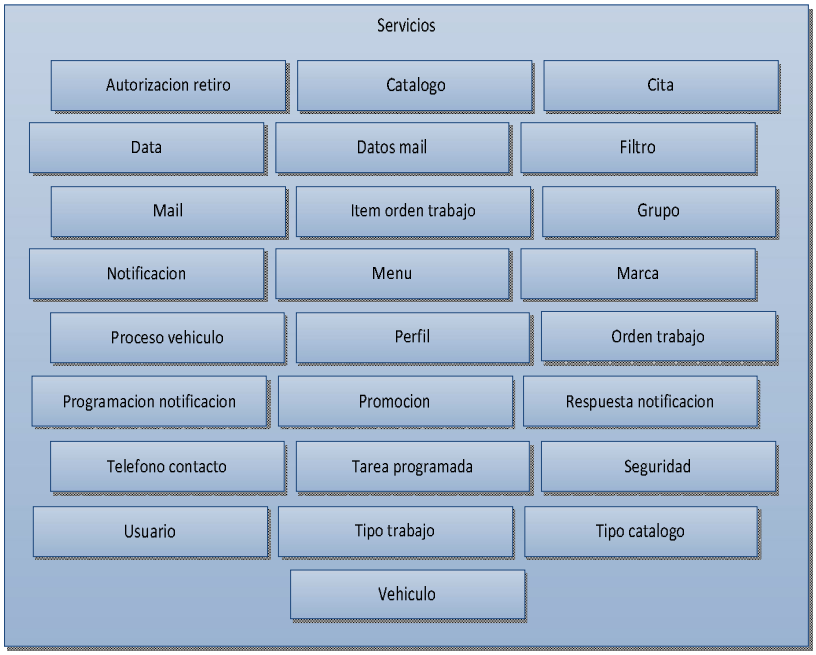
```
@Service
@Transactional
public class OrdenTrabajoService  extends FacadeCrudService<OrdenTrabajoDTO, Long>
implements IOrdenTrabajoService{
 //Código omitido

 @Transactional(readOnly = true)
 @Override
 public OrdenTrabajoDTO obtenerOrdenTrabajoDetallesPorId(Long idOrdenTrabajo) {
  return ordenTrabajoGestor.obtenerOrdenTrabajoDetallesPorId(idOrdenTrabajo);
 }
}
```

Elaborado por: Paul Jara & Javier Rivera.

También se hace uso del patrón de diseño fachada (Facade) para evitar escribir el mismo código para cada servicio o gestor que requiera la implementación de interfaces similares en estructura y funcionalidad, gracias a los tipos genéricos se hace posible realizar acciones genéricas con tipos de datos desconocidos.

En la figura 50 se muestra los servicios creados, y que permiten realizar las funcionalidades y requerimientos del sistema, existe una instancia de cada uno de los servicios en la clase MecanicaFactory.

Figura 50. Estructura de servicios del sistema



Elaborado por: Paul Jara & Javier Rivera.

Gestores

Los gestores son las clases que contienen la lógica de negocio y validaciones, se encuentran contenidos en paquetes para agrupar las funcionalidades con las que interactúan o con las que se relacionan.

Al igual que los servicios estas clases tienen una anotación pero en este caso se usa la anotación @Component en la parte superior para indicar a Spring que deben ser colocados en el contenedor de dependencias y que no son objetos especializados, como los servicios.

Los gestores se encuentran agrupados en paquetes según su funcionalidad o según las entidades a las que afectan en la figura 51 se muestra la implementación del gestor para órdenes de trabajo en el cual esta implementado el método para desactivar ordenes de trabajo aquí se realizan varias de operaciones sobre una entidad antes de persistir sus datos.

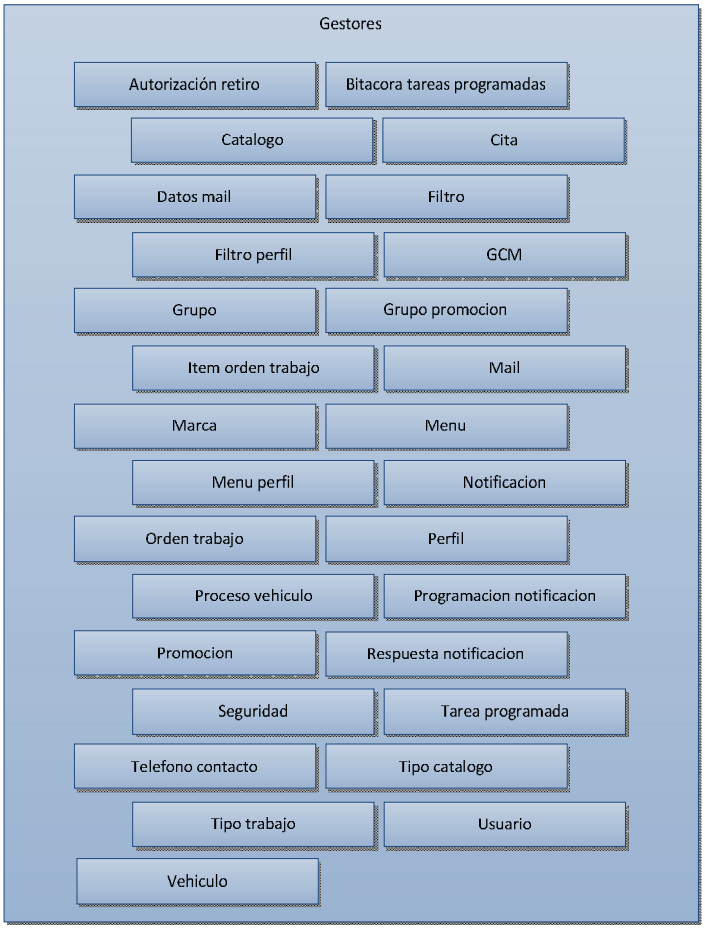
Figura 51. Gestor de órdenes de trabajo

```
@Component
public class OrdenTrabajoGestor extends FacadeCrudGestor<OrdenTrabajoDTO, Long>
  implements IOrdenTrabajoGestor {
//Código omitido
  @Override
  public void desactivarOrdenTrabajoSinItems(Long idProcesoVehiculo)
throws MecanicaException{
    try{
      Collection<Long> idsOrdenesTrabajo = obtenerIdsOrdenTrabajo(
idProcesoVehiculo);
      StringBuilder statment = new StringBuilder();
      statment.append("update OrdenTrabajoDTO ot set ot.activo = false
where ot.procesoVehiculo.id = :idProcesoVehiculo and ot.activo = true ");
      statment.append(idsOrdenesTrabajo.size() > 0 ? "and ot.id not in
(:idsOrdenesTrabajo)" : "");
      Query hquery =  dataGestor.createQuery(statment.toString())
          .setParameter("idProcesoVehiculo", idProcesoVehiculo);
        hquery.setParameterList("idsOrdenesTrabajo", idsOrdenesTrabajo);
      hquery.executeUpdate();
    } catch (Exception e) {
      throw new MecanicaException(e);
    }
  }
}
```

Elaborado por: Paul Jara & Javier Rivera.

En la figura 52 se muestra los gestores que componen la lógica de negocio de la capa de procesamiento, estos gestores interactúan entre ellos facilitando la reutilización de código y disminuyendo la viscosidad de código en los componentes de la arquitectura.

Figura 52. Estructura de los paquetes de gestores



Elaborado por: Paul Jara & Javier Rivera.

Spring maneja los beans entre gestores, y evita que se creen instancias de varias clases que contienen lógica por método que lo requiera, este mecanismo es realizado por el contenedor de dependencias de Spring este es un patrón de diseño conocido como Inversión de Control (IoC) y hace que el servidor responda más rápidamente ante una petición utilizando instancias de clases que se encuentran previamente creadas, optimizando el uso de recursos del servidor.

Spring permite la configuración desde archivos XML y desde anotaciones que se encuentran en las clases que lo requieran, estos dos tipos configuraciones trabajan en conjunto para obtener una aplicación modular y extensible.

Los archivos de configuración XML se encuentran bajo el directorio com/mecánica/laboteca/spring/config y se están segmentados según su funcionalidad de manera que sí se presenta un error en alguno de ellos es más fácil encontrar y corregir por el desarrollador.

Tareas programadas

Otra de las funcionalidades importantes del sistema son las *tareas programadas* que se ejecutaran durante la noche y se las puede configurar desde el archivo XML task-scheduler.xml, para su configuración se necesita conocer sobre el uso de expresiones de tiempo (CronTrigger expression), las cuales indicarán el momento del día y la frecuencia con la que se deben ejecutar las tareas programadas.

Ya que para las tareas programadas utilizadas en el sistema web se necesita de lógica de negocio ésta se encuentra en los gestores que son los encargados de realizar las acciones requeridas, y en caso de error durante la ejecución de las tareas que realicen algún cambio sobre la base de datos, el proceso completo realiza un retroceso a su estado anterior (Rollback), gracias a la utilización de Hibernate para la capa de persistencia y al uso de unidades persistencia para las transacciones de Java (JTA).

Figura 53. Configuración para envío de notificaciones diarias

```
<task:scheduler id="notificacionScheduler" pool-size="10" />
<task:scheduled-tasks scheduler="notificacionScheduler">
  <task:scheduled ref="tareaProgramadaSchedulerService"
    method="ejecutarTareaMailDiaria" cron="0 0 22 * * *" />
</task:scheduled-tasks>
```

Elaborado por: Paul Jara & Javier Rivera.

En la figura 53 se muestra la configuración para el envío de notificaciones diarias, el parámetro **pool-size** indica el número te hilos que pueden ejecutarse al mismo tiempo, para la tarea programada "notificacionScheduler", esta es utilizada en la configuración de la tarea mediante la etiqueta "<task:scheduled-tasks>" en la que se

indica que ejecutará el método "ejecutarTareaMailDiaria" todos los días a las 22 horas usando la expresión de tiempo "0 0 22 * * *".

Recursos

Los *recursos* que se encuentran segmentados por funcionalidad en la que se va a ser utilizado, y son accedidos desde la capa de procesamiento desde un archivo que apunta a la ubicación del archivo de propiedades.

DAO (Data Access Object)

Este patrón de diseño se utiliza para la interacción entre la capa de procesamiento con la capa de datos, se crea beans encargados de la consulta, inserción, actualización y borrado de registros de la base de datos de forma totalmente transparente del tipo de almacenamiento que se está empleando. Para la realización de las consultas se emplea Hibernate y Criteria Query, un API que facilita la composición de la consulta de una forma totalmente orientada a objetos.

4.1.2.3 Capa de acceso a datos.

Para construir la capa de persistencia se hace uso de Hibernate que permite mapear las tablas de la base de datos con las entidades de negocio y tener acceso a los datos en forma de objetos relacionales, abstrayendo al sistema Java de la base de datos con la que trabaja, también se hace uso de librerías que facilitan el realizar consultas.

Para mantener la integridad de datos JTA se encarga de crea una transacción por cada llamada a servicios que lleven la anotación @Transactional de manera que si durante la ejecución del servicio se produce un error, todas las operaciones realizadas contra la base de datos queden deshechas y se mantiene la integridad de datos.

Los **DTO** (data transfer object) son un tipo de objetos que sirven únicamente para transportar datos entre procesos

4.2 Funcionalidades

4.2.1 Persistencia.

Para persistir los datos se utiliza Hibernate, integrado con Spring que es la base del proyecto sobre la cual se construye las funcionalidades de negocio, garantizando la integridad de los datos mediante JTA y abstrayendo al sistema del motor de la base de datos.

Al ser Hibernate un proyecto de uso libre cuenta con gran cantidad de librerías creadas por la comunidad de software libre y otras fuentes, en el sistema se lo integra con la librería search-1.2.0 y search-hibernate-1.2.0 que facilitan la creación de consultas al proveer de interfaces fáciles y rápidas de utilizar una vez configurada la librería. Esta librería se la utiliza para la creación de componentes de búsqueda.

Se crean los beans con identificadores searchFacade y generalDAO que pueden ser inyectados en otros beans para hacer uso de las funcionalidades que proveen estos objetos.

Los archivos de configuración de Hibernate y su integración con Spring son varios entre los más importantes se encuentran generic-dao-hibernate.xml y datasource-tx-hibernate.xml aquí se especifica los métodos sobre los cuales se aplican la configuración orientada a aspectos que afectará a las clases y métodos que cumplen con los filtros especificados.

En la figura 54 se puede observar la configuración del bean searchFacade, en la cual se inyecta mediante referencia al objeto sessionFactory, este permite que la clase tenga acceso a la sesión de Hibernate que se encuentra en el contenedor de dependencias y pueda realizar operaciones con la base de datos.

En esta configuración también se especifica que para los métodos que coincidan con los parámetros indicados en la etiqueta <aop:pointcut> en el parámetro expression, se debe crear o agregar a la transacción llamada txAdviceHibernateSearchFacade.

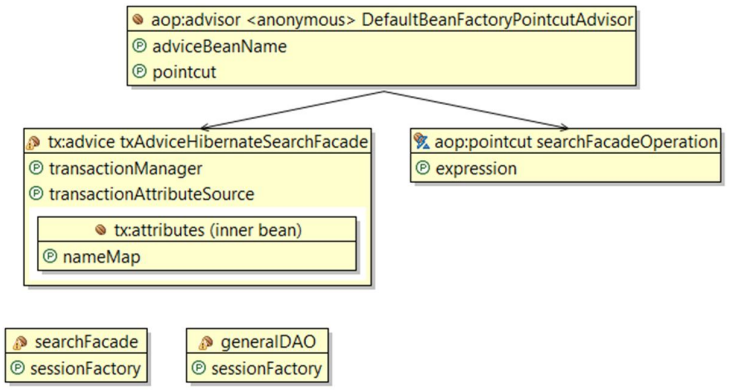
Figura 54. Configuración de librería generic-dao

```
<bean id="searchFacade"
  class="com.googlecode.genericdao.search.hibernate.HibernateSearchFacade">
  <property name="sessionFactory" ref="sessionFactory" />
</bean>

<aop:config>
  <aop:pointcut id="searchFacadeOperation"
    expression="execution(*
com.googlecode.genericdao.search.hibernate.HibernateSearchFacade.*(..))" />
  <aop:advisor pointcut-ref="searchFacadeOperation"
    advice-ref="txAdviceHibernateSearchFacade" />
</aop:config>
```

Elaborado por: Paul Jara & Javier Rivera.

En la figura 55 se muestra gráficamente la manera en la que se inyectan los beans para la configuración de la persistencia mediante programación orientada a aspectos.

Figura 55. Gráfico de AOP para las librerías generic-dao y search-facade



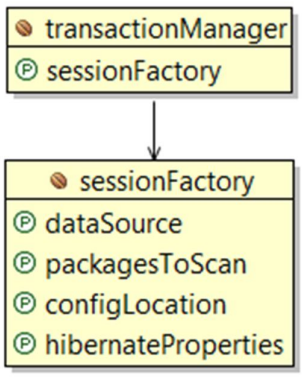
Elaborado por: Paul Jara & Javier Rivera.

Hibernate provee del objeto SessionFactory que da acceso a la sesión, a través de este objeto se puede realizar varios tipos de operaciones con la base de datos, este objeto es inyectado en otro objeto SessionFactoryBean que delega el manejo de las

transacciones a Spring, el cual se debe encargar de abrir y cerrar las transacciones cuando se requiera, además cuando el sistema se carga en el servidor este objeto permite realizar la validación de las entidades mapeadas y conocer si la base de datos se encuentra acorde a los DTO del sistema.

En la figura 56 se muestra gráficamente la manera en que se encuentra inyectado el bean sessionFactory en el objeto transactionManager que es una implementación de Spring para manejar las transacciones contra la base de datos en Java, facilitando de esta manera el desarrollo ya que se delega el comportamiento de las transacciones al marco de trabajo Spring.

Figura 56. Estructura de JTA de Spring



Elaborado por: Paul Jara & Javier Rivera.

4.2.2 Acceso a los servicios.

Para acceder a la capa de procesamiento utiliza el objeto MecanicaFactory que tiene la función de permitir el acceso desde la capa web a la capa de procesamiento, y crear los beans de servicio. En la figura 57 se muestra el método que inicializa el contexto al crear el objeto ClassPathXmlApplicationContext que recibe como parámetro en su constructor una lista de cadenas de texto que apuntan a los archivos de configuración XML. Al crear el contexto se mantiene esta instancia en la variable estática appCtx, a través de la cual se accede a los servicios del sistema.

Figura 57. Inicialización del contexto de la aplicación

```
Private static void init() {
    if (appCtx != null)
      return;

    String pathDataSource = "com/mecanica/laboteca/spring/config/datasource-jdbc.xml";
    String pathDataSourceTransaction = "com/mecanica/laboteca/spring/config/datasource-tx-
hibernate.xml";
    String pathGenericDaoHibernate = "com/mecanica/laboteca/spring/config/generic-dao-
hibernate.xml";
    String pathCollectionsSource = "com/mecanica/laboteca/spring/config/collections-source.xml";
    String pathAppContext = "com/mecanica/laboteca/spring/config/application-context.xml";
    String pathTaskScheduler = "com/mecanica/laboteca/spring/config/task-scheduler.xml";
    String pathMail = "com/mecanica/laboteca/spring/config/mail.xml";

    appCtx = new ClassPathXmlApplicationContext(new String[] {
       pathAppContext, pathDataSource, pathDataSourceTransaction,
       pathGenericDaoHibernate, pathCollectionsSource,
       pathTaskScheduler, pathMail });
  }
```

Elaborado por: Paul Jara & Javier Rivera.

4.2.3 Integración.

La integración entre el sistema web y la aplicación móvil se realiza mediante servicios web desarrollado con arquitectura REST, esta arquitectura que se centra en la optimización de los recursos del sistema además de ofrecer ventajas como el exponer URLs en forma de directorios para un rápido acceso, además transfiere los datos en formatos XML, JavaScript Object Notation (JSON), o ambos; es muy utilizada actualmente debido a su fácil acoplamiento con los diferentes frameworks, por su simplicidad para ser implementada y su estabilidad en la ejecución de peticiones y respuesta basadas en HTTP.

4.3 Revisión de planes de versión

4.3.1 Análisis de procesos.

Según lo sugerido por la metodología Scrum, inicialmente se elaboró el Product Backlog del proyecto, en este se detalla las diferentes funcionalidades requeridas para el sistema, el módulo a la que cada una pertenece, la prioridad de la misma, el tiempo estimado, la iteración en la que fue distribuida y las horas pendientes por cada iteración.

Para determinar la prioridad de cada funcionalidad, se realizó una evaluación de los procesos de mayor importancia del sistema, para ser desarrollados inicialmente con la finalidad que puedan ser revisados detalladamente por el cliente. El rango considerado se encuentra entre 100 y 1000, siendo 100 la de mayor prioridad; de igual manera la clasificación de las funcionalidades en las diferentes iteraciones se realizó en base a la prioridad.

Tabla 29. Product Backlog

Módulo	**Funcionalidades**	**Prioridad**	**Tiempo Estimado**	**Iteración**	**1**	**2**	**3**	**4**
				Horas Pend.	**1262**	**1082**	**808**	**410**
	Almacenar la información en una base de datos	100	60		60	0	0	0
	Implementación de la estructura del proyecto	200	120		120	0	0	0
	Iteración 1				**180**	**0**	**0**	**0**
Gestión de clientes	Administración de la información del usuario.	200	74		74	74	0	0
Gestión de clientes	Administración de la información del vehículo.	250	50		50	50	0	0
Gestión de clientes	Ingreso del proceso del vehículo	300	30		30	30	0	0
Operaciones	Registro de órdenes de trabajo	350	55		55	55	0	0
Operaciones	Registro y edición de mecánicos	400	30		30	30	0	0
Operaciones	Actualización de tareas finalizadas	400	35		35	35	0	0
	Iteración 2				**180**	**274**	**0**	**0**
	Primer entregable				**454**			
Operaciones	Entrega del vehículo	500	15		15	15	15	0
Promociones y Notificaciones	Administración de promociones	400	60		60	60	60	0

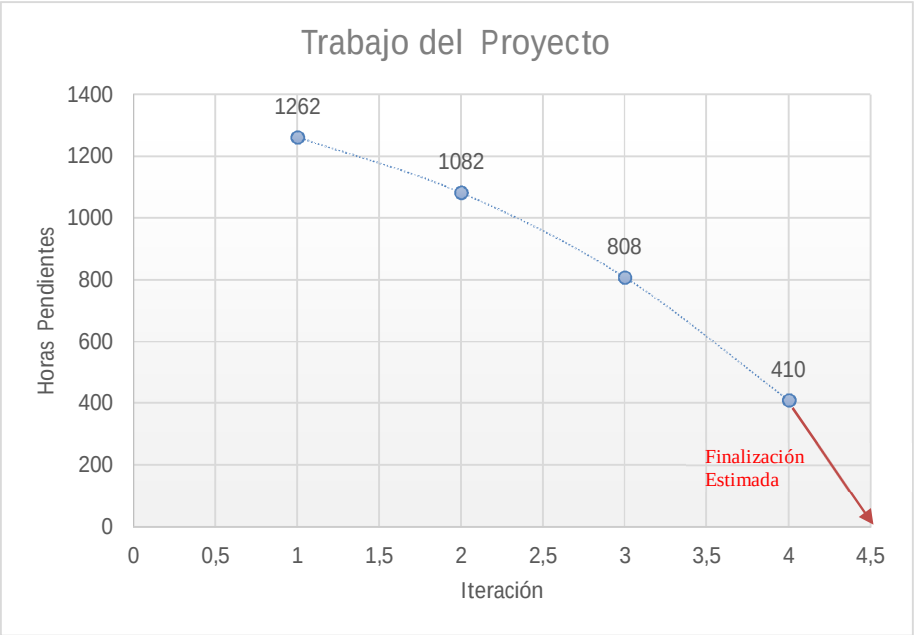
Promociones y Notificaciones	Presentación de promociones	450	25		25	25	25	0
Promociones y Notificaciones	Administración de notificaciones.	400	85		85	85	85	0
Promociones y Notificaciones	Gestión de envío de notificaciones.	450	65		65	65	65	0
Seguridad	Implementar seguridad del sistema web	400	68		68	68	68	0
Seguridad	Administración de perfiles	500	16		16	16	16	0
Seguridad	Administración de menús	500	8		8	8	8	0
Configuración	Administración de grupos	500	8		8	8	8	0
Configuración	Administración de catálogos	500	8		8	8	8	0
Configuración	Administración de catálogos	500	8		8	8	8	0
Configuración	Administración de marcas	500	8		8	8	8	0
Configuración	Administración de parámetros	500	8		8	8	8	0
Seguridad	Administración de filtro	500	16		16	16	16	0
	Iteración 3				**398**	**398**	**398**	**0**
Clientes	Consulta de información de vehículos (cliente)	500	20		20	20	20	20
Clientes	Consulta de notificaciones (cliente)	500	55		55	55	55	55
Clientes	Registro de solicitudes de citas (web)	500	70		70	70	70	70
Clientes	Administración de solicitudes de citas	500	35		35	35	35	35
Móvil	Conexión aplicativo móvil y servidor web	200	20		20	20	20	20
Móvil	Autenticación en el aplicativo móvil	300	25		25	25	25	25
Móvil	Consulta de promociones (móvil)	400	15		15	15	15	15
Móvil	Consulta de Notificaciones (móvil)	400	15		15	15	15	15
Móvil	Consulta de información vehículo (móvil)	400	30		30	30	30	30
Móvil	Registro solicitudes de citas (móvil)	200	30		30	30	30	30
Móvil	Notificaciones mediante mensajes inmediatos al aplicativo.	500	15		15	15	15	15
Móvil	Presentación del sistema	300	40		40	40	40	40
	Contratiempos	200	40		40	40	40	40
	Iteración 4				**410**	**410**	**410**	**410**
	Segundo entregable				**1262**			

Elaborado por: Paul Jara,& Javier Rivera

4.3.1.1 Gráficos de trabajo pendiente (Burndown charts)

Un gráfico de trabajo pendiente permite visualizar la velocidad con la que se están completando los requisitos a lo largo del tiempo desarrollo del proyecto. Es esencial para determinar si se puede cumplir con el objetivo en el tiempo estimado.

Figura 58. Burndown charts Product Backlog



Nota: En la figura se muestra las horas estimadas pendientes durante las diferentes iteraciones, como tiempo global se determinan 1262 horas que serán resueltas en un promedio de 315 horas por iteración, se plantean 4 iteraciones para concluir con el trabajo.

Elaborado por: Paul Jara & Javier Rivera.

4.4 Planificación y desarrollo del producto

La metodología Scrum sugiere la elaboración de un primer Sprint, que contenga las funcionalidades de mayor valor para el cliente.

4.4.1 Sprint 1.

El presente Sprint tiene la finalidad de desarrollar las funcionalidades de la primera y segunda iteración según consta en el Product Backlog.

Para la planificación de los diferentes Sprints se llevó a cabo varias reuniones con el Product Owner (Ing. Patricio Idrobo, gerente administrativo de la mecánica); uno de los primeros puntos fue definir el equipo de trabajo, el cuál quedo conformado de acuerdo a la tabla 30.

Tabla 30. Equipo de trabajo Sprint 1

Rol	Persona	Tareas
Product Owner	Ing. Patricio Idrobo	Administración del proyecto desde la perspectiva del negocio.
Scrum Master	Paul Jara	Revisión que el proceso Scrum se lleve a cabo de acuerdo a lo planificado.
Team	Paul Jara, Javier Rivera	Desarrollo de las funcionalidades.

Elaborado por: Paul Jara & Javier Rivera.

4.4.1.1 Análisis.

De acuerdo al valor de cada uno de los módulos, se determinó al módulo de gestión de clientes y módulo de operaciones como los primeros a ser desarrollados, estos módulos involucran los casos de uso:

- Gestión de vehículos.
- Gestión de usuario.
- Gestión de cliente.
- Gestión de ingreso de vehículo.
- Registro de tareas realizadas.
- Gestión de órden de trabajo.
- Gestión de tareas.

Nota: La información detallada de los casos de uso se encuentra en el capítulo 3.

Continuando con los requerimientos de la metodología Scrum, se elaboró el Sprint Backlog de la primera iteración, un documento donde se detallan las tareas a realizarse, los tiempos estimados y el estado de cada tarea, los estados considerados son:

- **Pendiente:** cuando la tarea aún no ha sido comenzada.
- **En proceso:** cuando se está desarrollando la tarea.
- **Pausado:** en caso que se requiera alguna información adicional para continuar con el desarrollo.
- **Finalizado:** cuando se ha culminado con la tarea.

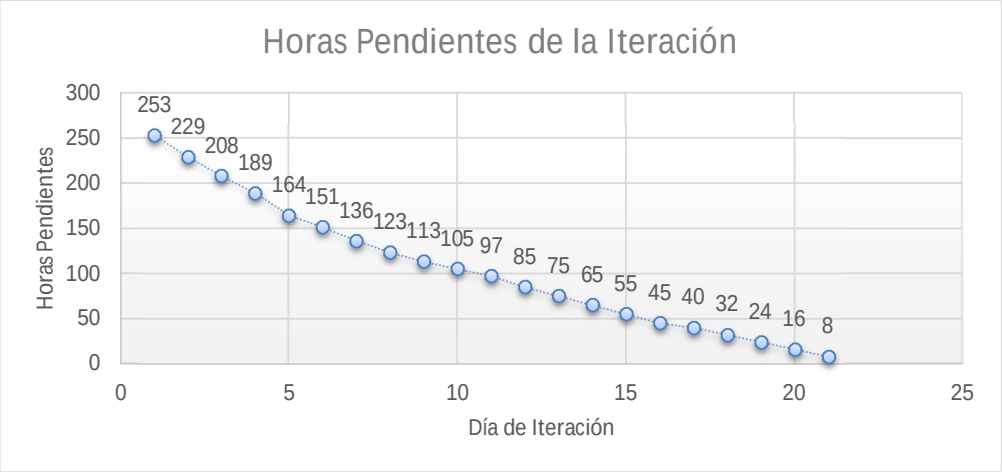
El documento será actualizado durante la ejecución de la iteración de acuerdo al avance de las tareas. Al finalizar la iteración se realizó otra reunión con el equipo de trabajo para determinar los resultados obtenidos, el estado de las tareas y el tiempo real de trabajo.

Tabla 31. Sprint Backlog 1

Iteración 1																																				
					Día	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Obs.
Tarea	**Prioridad**	**Encargado**	**Estado**	**Tiempo Estimado**	**Hora Pend**	202	198	178	158	139	122	104	90	66	46	24	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
					T Real																															
Modelado base de datos en Mysql	100	Paul	Finalizada	10	12	8	4																													
Mapeo de base de datos con Hibernate	200	Javier	Finalizada	50	52			10	10	12	10	10																								
Creación y Configuración estructura del proyecto (lógica del negocio)	100	Paul	Finalizada	40	52			10	10	7	7	5	5					8																		
Creación y Configuración estructura del proyecto (presentación)	200	Javier	Finalizada	40	49								5	12	10	12	10																			
Creación de componentes genéricos para búsqueda	300	Paul	Finalizada	40	45							3	4	12	10	10	6																			
Total				180	210	8	4	20	20	19	17	18	14	24	20	22	16	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Iteración 2																																				
					Día	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Obs.
	Prioridad	**Encargado**	**Estado**	**Tiempo Estimado**	**Hora Pend**	253	229	208	189	164	151	136	123	113	105	97	85	75	65	55	45	40	32	24	16	8	0	0	0	0	0	0	0	0	0	
					T. Real																															
Formulario y proceso para registrar, editar una persona adicional registrar como usuario	100	Paul	Finalizada	24	24	12	12																													
Formulario y proceso para asignar el perfil del usuario	200	Javier	Finalizada	15	17					7	3	7																								
Formulario y proceso para el ingreso de vehículo previa selección del usuario	250	Paul	Finalizada	25	25													5	10	10																
Formulario y proceso para la edición de asignación de vehículos con usuarios	300	Paul	Finalizada	15	15																10	5														
Formulario y proceso para la asignación de grupos al usuario	350	Javier	Finalizada	15	12				6	6																										
Formulario y proceso para registro de ingreso de vehículo al taller (creación de un proceso sobre el vehículo)	400	Paul	Finalizada	25	26			8	10	8																										
Formulario y proceso para el registro y modificación de teléfonos para los usuarios	500	Javier	En Proceso	15	13			10	3																											
Formulario y proceso para la creación de las órdenes de trabajo previa selección del proceso	450	Paul	Finalizada	25	30					4	10	8	8																							
Formulario y proceso para la asignación de ítems de trabajo a la orden de trabajo	500	Javier	Finalizada	25	27	12	12	3																												
Formulario y proceso para la asignación de mecánicos a cada orden de trabajo	550	Paul	Finalizada	25	23								5	10	8																					
Formulario y proceso para la actualización de tareas y cambio a las diferentes etapas del proceso	600	Paul	Finalizada	25	25											8	12	5																		
Pruebas de los procesos	200	Todos	Finalizada	40	40																		8	8	8	8	8									
				274	277	24	24	21	19	25	13	15	13	10	8	8	12	10	10	10	10	5	8	8	8	8	8	0	0	0	0	0	0	0	0	

Elaborado por: Paul Jara & Javier Rivera.

Figura 59. Burndown charts Sprint 1



Nota: En la figura se presenta gráficamente el avance del desarrollo de los requerimientos día a día, para el primer Sprint se estimaron 274 horas, que en tiempo real fueron 277 con una variación de 3 horas, las cuales fueron solventadas en un transcurso de 22 días. Para el ajuste de las horas faltantes se trabajó en varios días más del tiempo planificado.

Elaborado por: Paul Jara & Javier Rivera.

4.4.2 Sprint 2.

Este Sprint tiene la finalidad de desarrollar las funcionalidades de los módulos de promociones y notificaciones, seguridad, y configuración.

Para la planificación del Sprint 2 se llevó a cabo una reunión con el Product Owner (Ing. Patricio Idrobo), donde se definió el equipo de trabajo detallado a continuación.

Tabla 32. Equipo de trabajo Sprint 2.

Rol	Persona	Tareas
Product Owner	Ing. Patricio Idrobo	Administración del proyecto desde la perspectiva del negocio.
Scrum Master	Paul Jara	Revisión que el proceso Scrum se lleve a cabo de acuerdo a lo planificado.
Team	Paul Jara, Javier Rivera	Desarrollo de las funcionalidades.

Elaborado por: Paul Jara & Javier Rivera.

4.4.2.1 Análisis.

Continuando con la prioridad de las funcionalidades, para el desarrollo del tercer Sprint se estimó los casos de uso:

- Gestión de perfiles.
- Administración de notificaciones.
- Gestión de notificaciones.
- Gestión de tipo de trabajo.
- Gestión de promociones.
- Gestión de acceso.

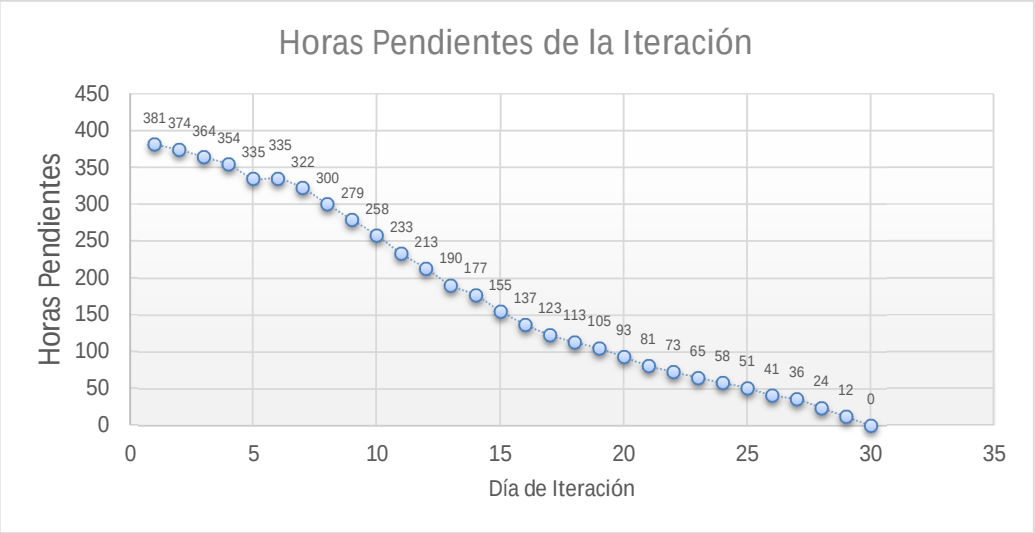
Nota: La información detallada de los casos de uso se encuentra en el capítulo 3.

Tabla 33. Sprint Backlog 2.

Tarea	Priori.	Encarg.	Estado	T. Estim	Día	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
					Pend.	318	374	364	354	335	335	322	300	279	258	233	213	190	177	155	137	123	113	105	93	81	73	65	58	51	41	36	24	12	0
					Tiem.																														
Formulario y proceso para la creación y edición de promociones	100	Paul	Fin.	30	29	12	7		10																										
Formulario y proceso para la asignación de promociones a grupos de usuarios	200	Javier	Fin.	25	27			10		12		5																							
Formulario y proceso para la creación y edición de notificaciones	100	Paul	Fin.	30	25					7		5	5					8																	
Formulario y proceso para la asignación de las notificaciones a grupos y /o usuarios	200	Paul	Fin.	25	25								5		10	5	5																		
Formulario y proceso para la creación, asignación y edición de tareas programadas a las notificaciones	200	Paul	Fin.	30	33							3	2	12		10	6																		
Formulario y proceso para él envió de notificaciones inmediatas	300	Javier	Fin.	20	22								10	9	3																				
proceso de envió de notificaciones según la programación	300	Paul	Fin.	35	36													4	10	12	10														
proceso para la visualización en las promociones en el sistema web	500	Paul	Fin.	20	20																	10	10												
Formulario y proceso para ingreso en el sistema	200	Paul	Fin.	8	8																			8											
Proceso de autorización y autenticación al sistema (Seguridad)	200	Paul	Fin.	40	40																				12	12	8	8							
CRUD de perfiles	500	Javier	Fin.	8	8										8																				
CRUD de menús	500	Javier	Fin.	8	7											7																			
CRUD de grupos	500	Javier	Fin.	8	7											3	4																		
CRUD de catálogos	500	Paul	Fin.	8	7																								7						
CRUD de tipos de catálogos	500	Javier	Fin.	8	8												5	3																	
CRUD de marcas	500	Javier	Fin.	8	8													8																	
CRUD de parámetros	500	Javier	Fin.	8	9														3	6															
CRUD de filtro	500	Javier	Fin.	8	8															4	4														
proceso para la asignación de menú a perfil	600	Paul	Fin.	8	7																									7					
proceso para la asignación de filtro a perfil	600	Javier	Fin.	8	8																4	4													
Formulario y funcionalidad para la entrega del vehículo y finalización del proceso de vehículo.	300	Paul	Fin.	15	15																										10	5			
Pruebas de los procesos	100	Todos	Fin.	40	36																												12	12	12
Total				**398**	**393**	**12**	**7**	**10**	**10**	**19**	**0**	**13**	**22**	**21**	**21**	**25**	**20**	**23**	**13**	**22**	**18**	**14**	**10**	**8**	**12**	**12**	**8**	**8**	**7**	**7**	**10**	**5**	**12**	**12**	**12**

Elaborado por: Paul Jara & Javier Rivera.

Figura 60. Burndown charts Sprint 2



Nota: En la figura se detalla gráficamente las horas solventadas diariamente hasta concluir las 381 horas estimadas para el segundo Sprint. Se empleó 30 días para la resolución de los requerimientos planificados en este Sprint.

Elaborado por: Paul Jara & Javier Rivera.

4.4.3 Sprint 3.

Este Sprint tiene la finalidad de desarrollar las funcionalidades de los módulos de gestión de reportes y web para clientes, así como todos los procesos correspondientes a la aplicación móvil.

Para la planificación del Sprint 3 se llevó a cabo una reunión con el Product Owner (Ing. Patricio Idrobo), en la misma que se definió el equipo de trabajo detallado en la tabla 34.

Tabla 34. Equipo de trabajo Sprint 3

Rol	Persona	Tareas
Product Owner	Ing. Patricio Idrobo	Administración del proyecto desde la perspectiva del negocio.
Scrum Master	Paul Jara	Revisión que el proceso Scrum se lleve a cabo de acuerdo a lo planificado.
Team	Paul Jara, Javier Rivera	Desarrollo de las funcionalidades.

Elaborado por: Paul Jara & Javier Rivera.

4.4.3.1 Análisis

Continuando con la prioridad de las funcionalidades, para el desarrollo del tercer Sprint se estimó los casos de uso:

- Gestión de reportes.
- Aplicación móvil.
- Consulta datos cliente.

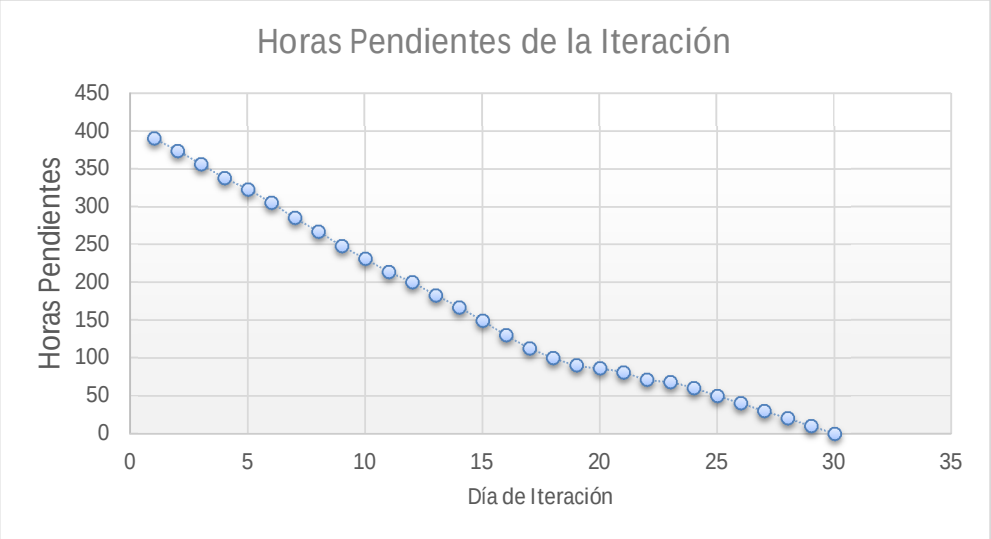
Nota: La información detallada de los casos de uso se encuentra en el capítulo 3 y 4.

Tabla 35. Sprint Backlog 3

Tarea	Prioridad	Encargado	Estado	T. Estim	Día	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
					Pend.	390	374	356	338	323	305	285	267	248	231	214	200	183	167	149	130	113	100	90	86	81	71	68	60	50	40	30	20	10	0
					Tiem																														
Formulario y proceso para la consulta de vehículos (cliente)	100	Paul	Fin.	20	22	4	8	10																											
Formulario y proceso para la consulta del citas (cliente)	100	Javier	Fin.	25	28	8	8	8	4																										
Formulario y proceso para el registro de solicitud de cita. (cliente)	100	Paul	Fin.	20	26				10	8	8																								
Formulario y proceso para la confirmación de la cita (cliente)	200	Paul	Fin.	25	25												5	10	8	2															
Formulario y proceso para la consulta del notificaciones (cliente)	100	Javier	Fin.	25	21				4	2	10	5																							
Formulario y proceso para el ingreso de respuesta a notificación (cliente)	100	Paul	Fin.	30	32							10	8	8	6																				
Formulario para la revisión y respuesta de las solicitudes de cita	300	Paul	Fin.	35	35															8	12	10	5												
Proceso de conexión de mediante un web service entre la aplicación móvil y el servidor web.	100	Paul	Fin.	20	19										4	10	5																		
Formulario y proceso para autenticación en la aplicación móvil	200	Javier	Fin.	25	25					5		5	10	5																					
Formulario y proceso para la consulta de promociones (móvil)	200	Javier	Fin.	15	15									6	7	2																			
Formulario y proceso para la consulta de notificaciones (móvil)	200	Javier	Fin.	15	16											5	4	7																	
Formulario y proceso para la consulta de la información del vehículo (móvil)	200	Javier	Fin.	15	16														8	8															
Formulario y proceso para la consulta procesos (móvil)	300	Paul	Fin.	15	15																						4	3	8						
Formulario y proceso para el ingreso de citas (móvil)	200	Paul	Fin.	15	15																		5	10											
Formulario y proceso para la confirmación de la cita para el cliente (móvil)	200	Javier	Fin.	15	17																7	7	3												
Proceso para el envío y recepción de mensajes Push.	200	Paul	Fin.	15	15																				4	5	6								
Pruebas de los procesos.	300	Todos	Fin.	40	20																									10	10				
Proceso para aplicación de estilos sobre el sistema.	200	Paul	Fin.	40	40																											10	10	10	10
Total				**410**	**402**	**12**	**16**	**18**	**18**	**15**	**18**	**20**	**18**	**19**	**17**	**17**	**14**	**17**	**16**	**18**	**19**	**17**	**13**	**10**	**4**	**5**	**10**	**3**	**8**	**10**	**10**	**10**	**10**	**10**	**10**

Elaborado por: Paul Jara & Javier Rivera.

Figura 61. Burndown charts Sprint 3



Nota: En la figura se puede apreciar el avance del desarrollo de los requerimientos planificados para el tercer Sprint, al igual que en las anteriores iteraciones la variación entre las horas estimadas y las horas reales es baja. Para este Sprint se planificaron 410 y las horas reales fueron de 402. Los días empleados para el desarrollo fueron de 30.

Elaborado por: Paul Jara & Javier Rivera.

Un punto importante de usar gráficos de trabajos pendientes es realizar simulaciones, considerando los cambios que sufriría el proyecto en las fechas de entrega si se le agregan o quitan requerimientos o integrantes en los equipos de trabajo.

CAPÍTULO 5
PRUEBAS DE FUNCIONAMIENTO

Las pruebas consisten en el proceso de evaluación del desempeño de las funcionalidades de la aplicación de software, con el objetivo de determinar la estabilidad y robustez que la aplicación ofrece al usuario.

5.1 Pruebas de stress

Una prueba de stress forza a un sistema informático al máximo punto, para medir el rendimiento, la capacidad y las condiciones bajo las cuales responde de manera óptima a las solicitudes de los usuarios. Es importante mencionar que el rendimiento está íntimamente ligado al hardware de la computadora sobre la cual se está ejecutando.

5.1.1 Software y hardware utilizados para las pruebas.

Para las pruebas de stress se utilizó JMeter, una aplicación open source desarrollada en Java, que está diseñada para probar el comportamiento funcional y medir el desempeño de una aplicación web.

Como se describió en el capítulo 4, se utilizó el servidor web JBoss 7.1 instalado sobre el sistema operativo Windows 7 Professional 64 bits Service pack 1.

El hardware que se utilizó fue un equipo Intel Core I7 3° generación, 12 Gigas en RAM con 256mb en video.

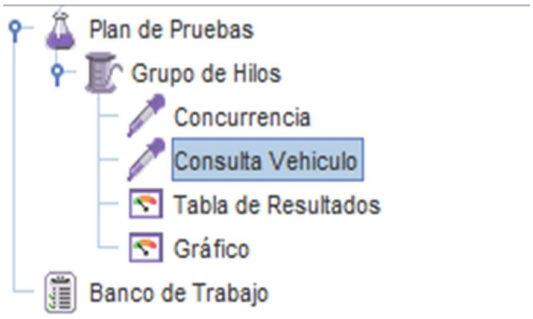
5.1.2 Pruebas.

Para las pruebas se tomó en consideración 3 parámetros, el número de usuarios concurrentes que soporta el sistema, el tiempo promedio de proceso de las peticiones y el porcentaje de error de cada uno de los casos.

Los procesos a ser evaluados son:

- Concurrencia.
- Consultar información del cliente.

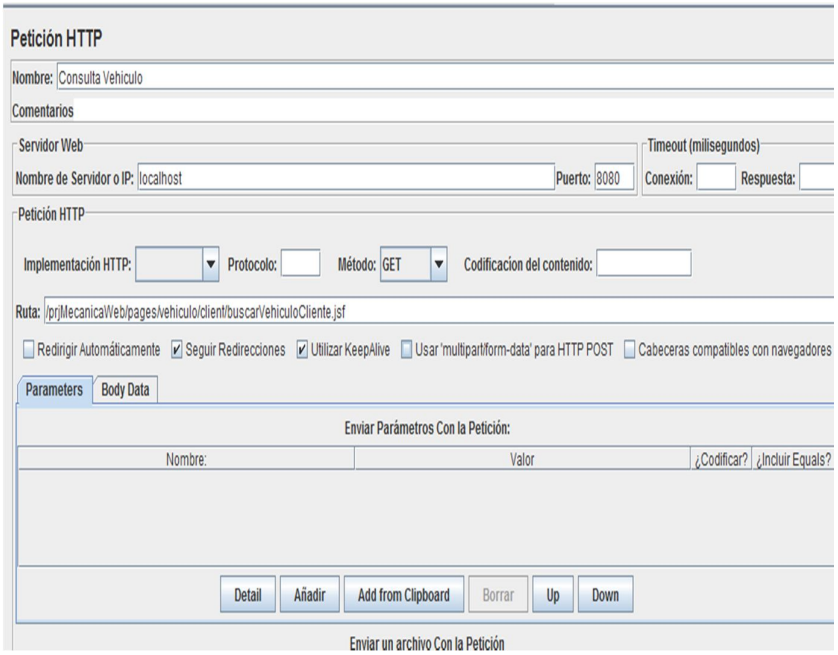
Figura 62. Plan de pruebas en JMeter



Elaborado por: Paul Jara & Javier Rivera.

Para cada una de las pruebas se conserva un lapso de tiempo de 2 segundos para realizar la siguiente petición, con la finalidad de dar realismo a la prueba.

Figura 63. Configuración de la petición



Elaborado por: Paul Jara & Javier Rivera.

5.1.3 Resultados.

En la figura 64 se muestra el resultado obtenido por cada petición, se detalla fecha hora de envió, nombre, el tiempo en milisegundos de respuesta, estado, número de bytes enviados y latencia o retardo, de estos datos se concluye que para 100 peticiones en un segundo, la aplicación responde satisfactoriamente a todas y en un tiempo aceptable, con un retardo de 1 a 2 milisegundos lo que es relativamente bajo. Considerando que la demanda estimada para la aplicación seria de máximo 50 usuarios por segundo, se podría operar correctamente con la configuración y equipo elegidos.

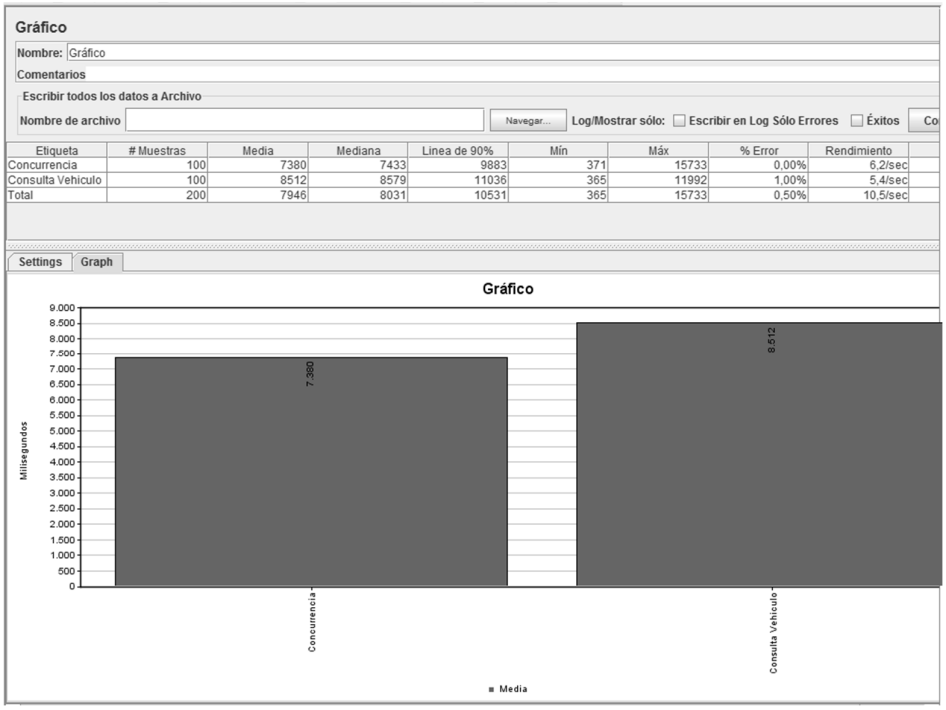
Figura 64. Tabla de resultados de las peticiones

Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Mues...	Estado	Bytes	Latency
167	17:33:26.840	Grupo de Hilos 1-71	Consulta Vehiculo	9016		10725	1
168	17:33:26.801	Grupo de Hilos 1-72	Consulta Vehiculo	9092		10725	1
169	17:33:26.761	Grupo de Hilos 1-73	Consulta Vehiculo	9171		10723	1
170	17:33:26.723	Grupo de Hilos 1-74	Consulta Vehiculo	9248		10725	1
171	17:33:26.684	Grupo de Hilos 1-75	Consulta Vehiculo	9324		9520	2
172	17:33:26.645	Grupo de Hilos 1-76	Consulta Vehiculo	9402		9518	2
173	17:33:26.573	Grupo de Hilos 1-77	Consulta Vehiculo	9512		10726	1
174	17:33:26.534	Grupo de Hilos 1-78	Consulta Vehiculo	9590		10726	2
175	17:33:26.478	Grupo de Hilos 1-79	Consulta Vehiculo	9684		10725	2
176	17:33:26.438	Grupo de Hilos 1-80	Consulta Vehiculo	9763		9519	1
177	17:33:26.383	Grupo de Hilos 1-81	Consulta Vehiculo	9855		9519	1
178	17:33:26.344	Grupo de Hilos 1-82	Consulta Vehiculo	9930		9520	1
179	17:33:26.291	Grupo de Hilos 1-83	Consulta Vehiculo	10023		10725	1
180	17:33:26.251	Grupo de Hilos 1-84	Consulta Vehiculo	10101		10725	1
181	17:33:26.198	Grupo de Hilos 1-85	Consulta Vehiculo	10192		10726	1
182	17:33:26.144	Grupo de Hilos 1-86	Consulta Vehiculo	10284		9518	1
183	17:33:26.104	Grupo de Hilos 1-87	Consulta Vehiculo	10362		10725	2
184	17:33:26.031	Grupo de Hilos 1-88	Consulta Vehiculo	10472		10726	2
185	17:33:25.991	Grupo de Hilos 1-89	Consulta Vehiculo	10550		10726	1
186	17:33:25.952	Grupo de Hilos 1-90	Consulta Vehiculo	10628		9518	2
187	17:33:25.909	Grupo de Hilos 1-91	Consulta Vehiculo	10709		10727	1
188	17:33:25.871	Grupo de Hilos 1-92	Consulta Vehiculo	10784		10727	2
189	17:33:25.831	Grupo de Hilos 1-94	Consulta Vehiculo	10883		10726	1
190	17:33:25.793	Grupo de Hilos 1-93	Consulta Vehiculo	10959		10726	2
191	17:33:25.755	Grupo de Hilos 1-95	Consulta Vehiculo	11036		10725	2
192	17:33:25.701	Grupo de Hilos 1-96	Consulta Vehiculo	11127		10726	1
193	17:33:25.662	Grupo de Hilos 1-97	Consulta Vehiculo	11203		9519	2
194	17:33:25.609	Grupo de Hilos 1-98	Consulta Vehiculo	11294		10725	2
195	17:33:25.569	Grupo de Hilos 1-99	Consulta Vehiculo	11372		10727	2
196	17:33:25.515	Grupo de Hilos 1-100	Consulta Vehiculo	11463		9519	1
197	17:33:25.438	Grupo de Hilos 1-3	Consulta Vehiculo	11640		10724	2
198	17:33:29.028	Grupo de Hilos 1-29	Consulta Vehiculo	8087		9519	1
199	17:33:25.475	Grupo de Hilos 1-9	Consulta Vehiculo	11693		10725	2
200	17:33:34.268	Grupo de Hilos 1-30	Consulta Vehiculo	3000		10726	2

Scroll automatically? Child samples? No. de Muestras 200 Última Muestra 3000 Media 7946 Desviación 2244

Elaborado por: Paul Jara & Javier Rivera.

Figura 65. Resultado gráfico de las pruebas



Elaborado por: Paul Jara & Javier Rivera.

Como se puede apreciar en los informes agregados que los tiempos de respuesta son bajos, ese tiempo está incluido el acceso a la BDD, por lo que se considera que el sistema está dentro de un rango aceptable.

5.2 Pruebas de caja negra.

Las pruebas de caja negra están enfocadas a evaluar el cumplimiento de los requisitos funcionales del sistema. A continuación se detalla la información obtenida de las pruebas realizadas sobre los procesos importantes de la aplicación web y móvil.

5.2.1 Módulo de seguridad.

Tabla 36. Pruebas de caja negra de gestión de acceso.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Ingresar*** sin datos ingresados	Presenta los mensajes: "Ingrese nombre de usuario", “Ingrese contraseña”.	Sí
2	Al ingresar los datos de autenticación incorrectos	Presenta un mensaje: "Usuario o contraseña Inválidos".	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Ingresar*** con datos correctos ingresados	Redirección a la pantalla de inicio con el menú correspondiente al usuario.	Sí
Conclusión	• La interfaz permite acceder al sistema solo con usuarios válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 37. Pruebas de caja negra ingreso y edición de usuarios.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta los mensajes: "Ingrese una identificación “,"Ingrese el nombre “,"Ingrese el apellido “,”e-mail no válido”.	Sí
3	Al ingresar los datos incorrectos en el cuadro número de documento.	Presenta un mensaje: “Número de cédula inválido”.	Sí
4	Al ingresar los datos incorrectos en el cuadro fecha.	Presenta un mensaje: "Fecha errónea”.	Sí
5	Al ingresar los datos incorrectos en el cuadro	Presenta un mensaje: "e-mail inválido”.	Sí

	email.		
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje:" ¿Desea grabar los cambios?" clic en aceptar graba los cambios, clic en el botón ***Cancelar*** cierra el mensaje. Presenta un mensaje: "Datos del usuario guardados correctamente"	Sí
Conclusión	• La interfaz permite registrar usuarios o modificarlos solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

5.2.2 Módulo de gestión de clientes.

Tabla 38. Pruebas de caja negra ingreso y edición de clientes.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros.	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta los mensajes: "Ingrese una identificación ","Ingrese el nombre ","Ingrese el apellido ","e-mail no válido".	Sí
3	Al ingresar los datos incorrectos en el cuadro número de documento.	Presenta un mensaje: "Número de cédula inválido".	Sí
4	Presionar el botón ***Asignar vehículo***.	Redirección al formulario de asignación de vehículo.	Sí
5	Presionar botón ***Teléfono***.	Redirección al formulario de ingreso de teléfonos.	Sí
Caso:	**Datos correctos**		
1	Presionar el botón **Guardar** con datos correctos ingresados.	Presenta un mensaje:" ¿Desea grabar los cambios?" clic en aceptar graba los cambios, clic en el botón ***Cancelar*** cierra el mensaje. Presenta un mensaje: "Datos del cliente	Sí

		guardados correctamente.”.	
Conclusión	• La interfaz permite registrar usuarios o modificarlos solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 39. Pruebas de caja negra ingreso y edición de vehículos.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros.	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta los mensajes:” Error de validación: se necesita un valor.”,” Ingrese la placa del vehículo”, “Ingrese el modelo del vehículo”, “Ingrese el kilometraje del vehículo”, “Ingrese el año del vehículo”, “Ingrese el color del vehículo”.	Sí
3	Ingresar texto erróneo en los cuadros año y kilometraje.	No escribe, solo permite el ingreso de números.	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje:” ¿Desea grabar los cambios?” clic en aceptar graba los cambios, clic en el botón cancelar cierra el mensaje. Presenta un mensaje: “Vehículo registrado correctamente.”.	Sí
Conclusión	• La interfaz permite registrar vehículos o modificarlos solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información		

Elaborado por: Paul Jara & Javier Rivera.

5.2.3 Módulo de operaciones.

Tabla 40. Pruebas de caja negra ingreso y edición proceso de vehículo.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Ingresar*** sin datos ingresados.	Presenta los mensajes "Elija un tipo de proceso; Debe registrar la fecha de ingreso; No hay información sobre el propietario del vehículo."	Sí
3	Ingresar fecha de ingreso erronea.	Presenta el mensaje "La fecha debe ser menor o igual a la actual."	Sí
4	Presionar link ***Buscar propietario.***	Despliega un formulario con filtros de búsqueda de los clientes ingresados, además permite ingresar un nuevo cliente.	Sí
5	Presionar link ***Buscar vehículo***.	Despliega un formulario con filtros para búsqueda de vehículos asignados al cliente.	Sí
5	Presionar el botón ***Nuevo***.	Limpia el formulario .	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje: "Proceso vehículo guardado correctamente".	Sí
2	Presionar ***Ingresar ítem***.	Redirección al formulario de ingreso de ítems.	Sí
3	Presionar ***Ingresar autorización***.	Redirección al formulario de ingreso de autorización.	Sí
4	Presionar ***Ver autorizaciones***.	Redirección al formulario de consulta de autorización.	Sí
5	Presionar el botón ***Crear orden de trabajo*.**	Redirección al formulario de creación de orden de trabajo.	Sí
Conclusión	• La interfaz permite agregar o modificar el proceso de vehículo solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 41. Pruebas de caja negra Ítems del proceso de vehículo.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros.	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Ingresar ítem*** sin datos ingresados.	Presenta el mensaje "Valor requerido." junto a los controles de categoría y tipo de trabajo.	Sí
3	Ingresar caracteres alfabéticos en cantidad.	Solo permite ingresar caracteres numéricos.	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje: "Ítem agregado".	Sí
Conclusión	• La interfaz permite agregar o modificar los ítems del proceso de vehículo solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 42. Pruebas de caja negra ingreso y edición de la autorización de retiro

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta el mensaje "Número de cédula inválido".	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje: "Autorización creada correctamente".	Sí
2	Presionar botón ***Ver autorizaciones***.	Redirección a un formulario con la tabla de las autorizaciones correspondientes al proceso.	Sí
Conclusión	• La interfaz permite agregar o modificar las autorizaciones de retiro del vehículo		

	solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 43. Pruebas de caja negra ingreso y edición de órdenes de trabajo

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Realizar búsqueda por filtros.	Presenta una tabla con los resultados coincidentes.	Sí
2	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta el mensaje "Elija el encargado de la orden".	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje: "Datos de la orden de trabajo guardados correctamente".	Sí
2	Presionar botón ***Asignar ítems***.	Muestra un formulario con los ítems disponibles para asignación.	Sí
Conclusión	• La orden de trabajo se relaciona directamente con la orden que se está creando. • La interfaz permite agregar o modificar las órdenes de trabajo del proceso de vehículo solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

5.2.4 Módulo de promociones y notificaciones.

Tabla 44. Pruebas de caja negra ingreso modificación de promociones.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Guardar*** sin datos	Presenta el mensaje "Error de validación: se necesita un valor.".	Sí

2	Ingresar fecha final de vigencia inferior a la fecha del sistema.	Presenta el mensaje, “La fecha debe ser mayor o igual a la fecha actual”.	Sí
3	Ingresar fecha inicio mayor a fecha de fin de vigencia.	Presenta un mensaje “fecha de inicio debe ser menor o igual a la fecha fin”.	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje:” ¿Desea grabar los cambios?” clic en aceptar graba los cambios y presenta un mensaje: “Promoción guardada exitosamente”, clic en el botón cancelar cierra el mensaje.	Sí
Conclusión	• La interfaz permite registrar promociones o modificarlas solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 45. Pruebas de caja negra ingreso modificación de notificaciones.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Guardar*** sin datos ingresados.	Presenta los mensajes: “Error de validación: se necesita un valor.”, “Ingrese un nombre”, “Ingrese un cuerpo”.	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje:” ¿Desea grabar los cambios?” clic en aceptar graba los cambios y presenta un mensaje: “Notificación guardada exitosamente”, clic en el botón cancelar cierra el mensaje.	Sí
Conclusión	• La interfaz permite registrar notificaciones o modificarlas solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

Tabla 46. Pruebas de caja negra ingreso modificación de tareas programadas.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Asignar Destino*** sin datos ingresados.	Presenta los mensajes: ”Hacen falta datos para este tipo de tarea”.	Sí
2	Ingresar fecha de inicio inferior a la fecha del sistema.	No permite el calendario se encuentra desactivado para fechas inferiores a la fecha de sistema.	Sí
3	Ingresar fecha inicio mayor a fecha de fin de vigencia.	Presenta un mensaje “fecha de inicio debe ser menor o igual a la fecha fin”	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Guardar*** con datos correctos ingresados.	Presenta un mensaje: “La tarea programada se guardó correctamente.”.	Sí
Conclusión	• La interfaz permite agregar o modificar tareas programadas solo con datos válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

5.2.5 Módulo de información para dispositivos móviles.

Tabla 47. Pruebas de caja negra acceso al aplicativo móvil.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Ingresar*** sin datos ingresados.	Presenta los mensajes: "Ingrese usuario y contraseña”.	Sí
2	Al ingresar los datos de autenticación incorrectos.	Presenta un mensaje: "Usuario o contraseña no válidos".	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Ingresar*** con datos correctos ingresados.	Redirección a la pantalla de menú	Sí
Conclusión	• La interfaz permite acceder al aplicativo solo con usuarios válidos.		

	• La interfaz alerta al usuario sobre errores en el ingreso de información.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 48. Pruebas de caja negra ingreso de solicitud de cita.

Caso:	**Datos incorrectos**		
Num.	**Acción**	**Resultado**	**Éxito**
1	Presionar el botón ***Ingresar*** sin datos ingresados.	Presenta los mensajes: "Ingrese usuario y contraseña".	Sí
2	Al ingresar la fecha de solicitud inferior a la fecha actual.	Presenta un mensaje: "La fecha de solicitud debe ser mayor al menos en 2 horas a la fecha actual".	Sí
Caso:	**Datos correctos**		
1	Presionar el botón ***Ingresar*** con datos correctos ingresados.	Presenta un mensaje:" Se va a registrar una solicitud de cita ¿Desea continuar?" clic en aceptar graba los cambios y presenta el mensaje "Su petición fue ingresada correctamente", clic en el botón ***Cancelar*** cierra el mensaje.	Sí
Conclusión	• La interfaz permite acceder al aplicativo solo con usuarios válidos. • La interfaz alerta al usuario sobre errores en el ingreso de información.		

Elaborado por: Paul Jara & Javier Rivera.

CONCLUSIONES

- El uso de Spring como framework para el desarrollo del proyecto, brinda soporte tanto para la aplicación web al servir como contenedor de inversión, como para el aplicativo móvil que utiliza servicios web implementados con Spring Web Service, de esta manera se obtiene un sistema más robusto y mejor integrado.

- Scrum ofrece una metodología para el desarrollo de aplicaciones en grupos de trabajo de pocos integrantes, no así para grupos grandes, debido a la dificultad en el control de las tareas de los miembros del equipo.

- La integración del sistema web y el aplicativo móvil permite a los usuarios recibir información relevante de la mecánica, en la pantalla del dispositivo móvil, de forma ágil y cómoda, confirmando que la integración de los sistemas mediante servicios web y GCM es de gran importancia para alcanzar el objetivo planteado.

- La elección del sistema operativo Android como base para la aplicación móvil fue una decisión acertada, al ser el más usado actualmente en el país, ofrece la posibilidad de llegar a un más amplio número de clientes con el servicio.

- La automatización del envío de notificaciones periódicas a los clientes, es un plus de gran valor, que le permitirá a la mecánica dar a conocer sus servicios y promociones, en menor tiempo del que se usa actualmente para este fin.

- En la metodología utilizada para el desarrollo de las aplicaciones, la colaboración del cliente en todo el proceso fue muy importante, ya que su aporte enfocado en la lógica del negocio ayuda a una mejor comprensión de las necesidades por parte del equipo de trabajo.

- Utilizar herramientas open source en el desarrollo de las aplicaciones es de gran utilidad, debido a que evitan al usuario el pago de licencias y permiten su libre reproducción y distribución, pero tiene como desventaja que el soporte técnico oficial para estas puede ser inexistente o de difícil acceso.

RECOMENDACIONES

- En el desarrollo de una aplicación de software es necesario realizar un correcto levantamiento de requerimientos y necesidades del negocio, de esta forma se puede realizar un óptimo diseño y asegurar la satisfacción del cliente.

- Para la implementación del sistema es recomendable separar en módulos las dependencias utilizadas para levantar la aplicación en el servidor JBoss ya sea en la versión 7 o 6 EAP, esto con la finalidad de que las dependencias utilizadas por otra aplicación que se encuentre en el mismo servidor no se repita, facilitando la actualización de las dependencias usadas así como también la actualización del aplicativo en sí mismo.

- La capacitación del manejo de la aplicación al usuario administrador es muy importante para mantener el sistema trabajando eficientemente, existen varios parámetros que al ser cambiados incorrectamente pueden provocar la inestabilidad del sistema.

- Se recomienda incluir en el sistema un módulo de facturación que se pueda integrar con el sistema de contabilidad e inventarios que utilizan en la mecánica Romero Hnos. Laboteca, logrando así la información generada sea útil para ambos sistemas.

- Se recomienda implementar un portal que muestre información, de los servicios prestados por la mecánica así como también información y datos de contacto además que brinde acceso a la aplicación web para los clientes que deseen acceder la información de sus vehículos y demás utilidades que brinda el sistema web.

- Realizar varios tipos de pruebas a las aplicaciones es de gran importancia para ofrecer un producto de calidad al cliente, además permite determinar los ambientes óptimos en hardware y software para su ejecución.

LISTA DE REFERENCIAS

- *JBoss*. (01 de noviembre de 2012). Recuperado el 18 de mayo de 2014, de JBoss: http://docs.jboss.org/hibernate/orm/3.2/api/org/hibernate/Transaction.html

- *Oracle*. (08 de diciembre de 2012). Recuperado el 17 de mayo de 2014, de http://www.oracle.com/es/technologies/java/features/index.html

- Albaladejo, X. (01 de enero de 2012). *proyectosagiles*. Recuperado el 02 de febrero de 2013, de proyectosagiles: http://www.proyectosagiles.org/lista-requisitos-priorizada-product-backlog

- androidcurso. (07 de diciembre de 2011). *androidcurso*. Recuperado el 01 de agosto de 2013, de androidcurso: http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android

- Ávila, S. (10 de abril de 2009). *apliweb4.foroactivos*. Recuperado el 10 de febrero de 2013, de apliweb4.foroactivos: http://apliweb4.foroactivos.net/t1-conceptos-basicos-aplicaciones-web

- Ceresola Millet , J. C. (2012). *Material didáctico para la enseñanza de SPRING*. Valencia: etsing.

- Clarence, H., & Harrop, R. (2012). *Pro Spring 3*. New York: Apress.

- developer.android.com. (2012). *developer.android.com.* Recuperado el 5 de febrero de 2013, de developer.android.com: http://developer.android.com/Google/gcm/index.html

- Fuentes, V., & Guevara, J. (01 de enero de 2010). Análisis del patrón Modelo Vista Controlador implementado en lenguajes de software libre para el desarrollo de aplicaciones web. Caso práctico: Liceo de talentos Stephen Hawking. Riobamba, Chimborazo, Ecuador.

- Gómez, J. J. (01 de noviembre de 2011). Arquitectura MVC.

- Manami, D. (1 de enero de 2009). *ingenieriadesoftware.mex.* Recuperado el 1 de febrero de 2013, de ingenieriadesoftware.mex: http://ingenieriadesoftware.mex.tl/52812_Scrum.html

- Marchioni, F. (2011). JBoss AS 7. En F. Marchioni, *Configuration, Deployment, and Administration.*

- Marchioni, F. (4 de diciembre de 2012). *Mastertheboss*. Recuperado el 17 de mayo de 2014, de Mastertheboss: http://www.mastertheboss.com/jboss-as-7/jboss-as-7-introduction

- Mkyong. (25 de marzo de 2010). *Mkyong.com.* Recuperado el 18 de mayo de 2014, de Mkyong.com: http://www.mkyong.com/spring/spring-aop-examples-advice/

- Molina, O. (27 de junio de 2012). *slideshare.* Recuperado el 16 de mayo de 2013, de slideshare: http://www.slideshare.net/Onymariam-Molina/sistema-operativo-android-13470154

- Oracle. (30 de agosto de 2011). *Java*. Recuperado el 17 de mayo de 2014, de Java: http://www.java.com/es/about/

- Oracle. (08 de diciembre de 2011). *Java*. Recuperado el 17 de mayo de 2014, de Java: http://www.java.com/es/download/faq/techinfo.xml

- Otero, D. A. (01 de enero de 2013). *issuu.* Recuperado el 10 de noviembre de 2013, de issuu: http://issuu.com/aeegle/docs/aeegle_-__qu__es_cloud_computing-

- Río, M. M. (01 de enero de 2012). *QUE ES EL CLOUD COMPUTING.* Recuperado el 10 de febrero de 2013, de internetsano: http://www.internetsano.gob.ar/archivos/cloudcomputing_empresas.pdf

- Rodriguez, A. (06 de noviembre de 2008). *IBM.* Recuperado el 12 de diciembre de 2013, de IBM: http://www.ibm.com/developerworks/webservices/library/ws-restful/ws-restful-pdf.pdf

- Walls, C. (2011). *Spring in Action.* Shelter Island, NY: Manning.

GLOSARIO

A

Android SDK: Android SDK (Software Development Kit) es el conjunto de herramientas y librerías desarrolladas por Google para desarrollar, compilar y depurar aplicaciones para el sistema operativo Android

AOP: programación orientada al aspecto que nos ayuda a modificar dinámicamente nuestro modelo estático para incluir el código requerido para cumplir los requerimientos secundarios sin tener que modificar el modelo estático original.

API: una API (Interfaz de Programación de Aplicaciones) es un conjunto de funciones que permite al programador acceder a servicios de una aplicación a través del uso de un lenguaje de programación.

Aplicación móvil (APP): es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles.

Aplicación web: en la ingeniería de software se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.

AWT: permite hacer interfaces gráficas mediante artefactos de interacción con el usuario, como botones, menús, texto, botones para selección, barras de deslizamiento, ventanas de diálogo, selectores de archivos, etc.2

B

BDD: base de datos, colección de información almacenada de forma organizada en una computadora

Blue-ray: disco de almacenamiento óptico de 12 cm desarrollado por Blu-Ray Disc Association capaz de contener la gran cantidad de datos requeridos en películas de alta definición, además de otros actores inherentes.

C

C: es un lenguaje complejo inventado en los laboratorios Bell a comienzos de los 70 como una herramienta para programar sistemas operativos como UNIS.

C++: es una variante de sistema C que se beneficia de una metodología de programación moderna llamada programación orientada a objetos.

Caching: almacenar temporalmente los datos frecuentemente accedidos más cerca del solicitante de los mismos.

CSS (Hojas de Estilo en Cascada): apartados que tienen que ver directamente con el lenguaje utilizado para definir los estilos en páginas web.

F

Formato APK: un fichero APK es una variante del formato JAR de Java. Es un fichero en formato comprimido ZIP donde se ha empaquetado cuatro tipos de información: el código, los recursos, la firma digital y el fichero de manifiesto.

Framework: es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

G

GCM: (Google Cloud Messaging) es un servicio gratuito que permite que los desarrolladores envíen datos de los servidores a sus aplicaciones de Android. Puede tratarse de un mensaje corto que indique a la aplicación de Android que hay datos nuevos que se pueden recuperar del servidor (por ejemplo, una película subida por un amigo) o puede ser un mensaje que contenga hasta 4 KB de datos de carga.

Google Docs: es un sencillo pero potente procesador de texto y hoja de cálculo, todo en línea, que nos permite crear nuevos documentos, editar los que ya teníamos o compartirlos en la red con otros usuarios.

GPS: es un dispositivo que puede utilizar las señales de posicionamiento global para determinar su localización.

H

HQL (Hibernate Query Languaje): es el lenguaje de consultas que usa Hibernate para obtener los objetos desde la base de datos. Su principal particularidad es que las consultas se realizan sobre los objetos java que forman nuestro modelo de negocio.

I

iText: es una librería para java (aunque está disponible también para el lenguaje C#), que permite generar archivos pdf de una forma muy sencilla.

J

Java Cards: se refiere a una tecnología que permite a las aplicaciones basadas en Java que se ejecuten de forma segura en tarjetas inteligentes y pequeños dispositivos de huella de memoria similares

Java ME: esta es una plataforma de desarrollo de aplicaciones para dispositivos con recursos limitados, como por ejemplo: celulares, PDA's, Palm's, etc.

Java SE (Java Enterprise Edition): es la versión más grande de JAVA y se utiliza por lo general para crear aplicaciones grandes de cliente/servidor y para desarrollo de WebServices.

JDBC (Java Database Connectivity): es la parte de Java que nos va a permitir conectarnos con bases de datos relacionales utilizando el lenguaje SQL.

JDK: se trata de un paquete de software que puede utilizar para desarrollar aplicaciones basadas en Java. (Java)

JMS (Java Message Service): es un sistema de comunicación entre aplicaciones en base a mensajes, la comunicación puede realizarse de dos formas punto a punto (P2P) y Pub/Sub.

JNDI ("Java Naming Directory Interface"): es una especificación que permite localizar información en distintos directorios distribuidos.

JRE: es una máquina virtual de Java y su función es hacer de intermediario entre una aplicación programada en Java y el sistema operativo que se esté usando.

JSON formato para el intercambio de información a través de un archivo.

JTA (Java Transaction API): es un juego de librerías ("packages") utilizadas en el lenguaje Java para definir transacciones programáticamente, esto es, definir manualmente el uso de transacciones.

JUnit: es un entorno que permite ejecutar test de clases Java.

L

Latencia: es el lapso necesario para que un paquete de información viaje desde la fuente hasta su destino. La latencia y el ancho de banda, juntos, definen la capacidad y la velocidad de una red.

Lenguaje de programación: son estructuras simbólicas que nos permite disponer de los dispositivos de una computadora.

Linux: es un sistema operativo basado en UNIX, mantenido por voluntarios, y distribuido de forma gratuita. Linux se utiliza mayoritariamente en los servidores y computadoras incrustadas.

N

Navegador: se denomina navegador a un programa que sirve para mostrar páginas web. Un navegador funciona como un cliente al que un servidor HTTP sirve los documentos.

O

Oracle Corporation: es una compañía que nació bajo dicho nombre en 1983, fue conocida como Relational Software Inc. Su fundador Larry Ellison en 1977 esta empresa se llamaba Software Development Laboratories.

P

Plataforma: combinación de hardware y software de sistema operativo sobre el que se construye una aplicación.

POO: es un paradigma de programación que define los programas en términos de "clases de objetos"

R

RMI: es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales de Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas.

RPC: procedimiento remoto o Remote Procedure call (en inglés) es una poderosa técnica para construir aplicaciones cliente-servidor distribuidas.

S

Scrum: es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

Servidor de aplicaciones: unidad funcional que proporciona acceso en red a determinado tipo de software.

Sistema operativo: sistema de programas que lleva a cabo distintas operaciones técnicas, proporcionando una capa adicional de aislamiento entre el usuario y el mundo de bits y bytes representado por el hardware de la computadora.

Smartphone: es un teléfono móvil, pero tiene unas características diferentes, muchas de ellas propias de los ordenadores personales, por eso se les llama "teléfonos inteligentes".

Spring Web Service: es un producto de la comunidad de Spring centrado en la creación de servicios web basados en documentos, tiene como objetivo facilitar el desarrollo de servicios SOAP, lo que permite la creación de servicios web flexibles utilizando una de las muchas maneras de manipular cargas de XML.

Spring: es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

Sprint: es una iteración de un proceso de desarrollo de software basado en metodologías ágiles.

SQLite es un sistema de gestión de bases de datos relacional. Empleado para equipos de poca capacidad.

Sun Microsystem: fue una empresa informática que se dedicaba a vender estaciones de trabajo, servidores, componentes informáticos, software (sistemas operativos) y servicios informáticos.

Anexo 1. Diccionario de datos

Tabla 1. Tabla autorizacion_retiro

Descripción: contiene las autorizaciones de retiro del vehículo

Campo	Tipo	Nulo	Pk	Descripción
Id	int(11)	No	Si	Clave primaria de la tabla autorizacion_retiro.
id_proceso_vehiculo	int(11)	No	No	Clave foránea de la tabla proceso_vehiculo.
id_usuario	int(11)	Si	No	Clave foránea de la tabla usuario.
tipo_identificación	int(11)	Si	No	Almacena el tipo de identificación FK tabla catálogo.
nombre	varchar(250)	Si	No	Almacena el nombre de la persona a retirar el vehículo.
apellido	varchar(250)	Si	No	Almacena el apellido de la persona a retirar el vehículo.
numero_idetificacion	varchar(25)	Si	No	Almacena el número de identificación de la persona a retirar el vehículo.
es_usuario	bit(1)	Si	No	Determina si es usuario del sistema.
Utilizada	bit(1)	Si	No	Determina si se realizó el retiro del vehículo.
Activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 2. Tabla bitacora_tarea_programada

Descripción: Contiene la información de las tareas programadas ejecutadas

Campo	Tipo	Nulo	PK	Descripción
id	int(11)	No	Si	Clave primaria de la tabla bitacora_tarea_programada.
id_tarea_programada	int(11)	No	No	Clave foránea de la tabla tarea_programada.
fecha_ejecucion	Datetime	Si	No	Determina la fecha de ejecución.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 3. Tabla catalogo

Descripción: contiene información de catálogos

Campo	Tipo	Nulo	PK	Descripción
id	int(11)	No	Si	Clave primaria de la tabla.
id_tipo_catalogo	int(11)	No	No	FK de la tabla tipo_catalogo.
codigo_referencia	varchar(5)	Si	No	Almacena el código de referencia.
información	varchar(200)	Si	No	Almacena información del catálogo.
valor	varchar(60)	Si	No	Almacena el valor.
tipo_dato	int(11)	Si	No	Almacena el tipo de dato.

orden	int(11)	Si	No	Almacena el orden.
editable	bit(1)	No	No	Determina si es editable.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 4. Tabla catalogo_relacion

Descripción: contiene la relación entre catálogos				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla catalogo_relacion.
id_catalogo_padre	int(11)	No	No	FK de la tabla catalogo_padre.
id_catalogo_hijo	int(11)	No	No	FK de la tabla catalogo_hijo.
orden	int(11)	Si	No	Almacena el orden.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 5. Tabla cita

Descripción: contiene información de cita				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla cita.
id_usuario	int(11)	No	No	FK de la tabla usuario.
id_estado_cita	int(11)	Si	No	FK de la tabla estado_cita.
id_tipo_proceso	int(11)	Si	No	FK de la tabla tipo_proceso.
descripcion_peticion	varchar(500)	Si	No	Almacena la descripción de petición.
descripcion_respuesta	varchar(500)	Si	No	Almacena la descripción de respuesta.
fecha_requerida	Timestamp	Si	No	Almacena la fecha requerida.
fecha_confirmacion	Timestamp	Si	No	Almacena la fecha de confirmación.
versión	int(11)	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 6. Tabla filtro

Descripción: contiene información de los filtros de seguridad				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla filtro.
patrón	varchar(255)	No	No	Almacena el patrón.
orden	int(11)	Si	No	Almacena la orden.
descripción	varchar(255)	Si	No	Almacena la descripción
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 7. Tabla filtro_perfil

Descripción: contiene la relación entre filtro y perfil				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla filtro_perfil.
id_perfil	int(11)	Si	No	FK de la tabla perfil.
id_filtro	int(11)	No	No	FK de la tabla filtro.
expresion	varchar(255)	Si	No	Almacena la expresión.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 8. Tabla grupo

Descripción: contiene los datos de los grupos de usuarios				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla grupo.
nombre	varchar(60)	Si	No	Almacena el nombre.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 9. Tabla grupo_promocion

Descripción: contiene la relación entre los grupos y las promociones				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla grupo_promocion.
id_promocion	int(11)	No	No	FK de la tabla promocion.
id_grupo	int(11)	Si	No	FK de la tabla grupo.
general	bit(1)	Si	No	Determina la general.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 10. Tabla item_orden_trabajo

Descripción: contiene los ítems relacionados a la orden de trabajo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla item_orden_trabajo.
id_proceso	int(11)	No	No	FK de la tabla proceso.
id_tipo_trabajo	int(11)	No	No	FK de la tabla tipo_trabajo.
id_estado_item	int(11)	No	No	FK de la tabla estado_item.
id_item_precedente	int(11)	Si	No	FK de la tabla item_precedente.
id_orden_trabajo	int(11)	Si	No	FK de la tabla orden_trabajo.
cantidad	int(11)	Si	No	Almacena la cantidad.
tiempo_real	int(11)	Si	No	Almacena el tiempo real.
descripcion	mediumtext	Si	No	Almacena la descripción.
nota	mediumtext	Si	No	Almacena la nota.
activo	bit(1)	Si	No	Determina si el registro está activo.
version	int(11)	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 11. Tabla menu

Descripción: contiene la información del menú del sistema				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla menu.
id_menu_padre	int(11)	Si	No	FK de la tabla menu_padre.
accion	varchar(255)	Si	No	Almacena la acción.
url	varchar(255)	Si	No	Almacena la Url.
ant_pattern	varchar(1024)	Si	No	Almacena el id del padre.
nombre	varchar(80)	Si	No	Almacena el nombre.
modulo	int(11)	Si	No	Almacena el modulo.
orden	int(11)	Si	No	Almacena el orden.
nivel	int(11)	Si	No	Almacena el nivel.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 12. Tabla notificacion

Descripción: contiene la información de la notificación				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla notificación.
id_tipo_notificacion	int(11)	No	No	FK de la tabla tipo_notificacion.
nombre	varchar(60)	Si	No	Almacena el nombre.
cuerpo	varchar(5000)	Si	No	Almacena el cuerpo.

requiere_respuesta	bit(1)	Si	No	Almacena la requiere respuesta.
activo	bit(1)	Si	No	Determina si el registro está activo.
created_by	int(11)	Si	No	Campo para auditoría.
created_date	Timestamp	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 13. Tabla orden_trabajo

Descripción: contiene la información de la orden de trabajo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla orden_trabajo.
id_estado_orden_trabajo	int(11)	No	No	FK de la tabla estado_orden_trabajo.
id_proceso_vehiculo	int(11)	No	No	FK de la tabla proceso_vehículo.
id_usuario_asigno	int(11)	Si	No	FK de la tabla usuario_asigno.
id_usuario_encargado	int(11)	Si	No	FK de la tabla usuario_encargado.
fecha_ingreso	Datetime	Si	No	Almacena la fecha de ingreso.
Nota	mediumtext	Si	No	Almacena la nota.
Activo	bit(1)	Si	No	Determina si el registro está activo.
Versión	int(11)	Si	No	Campo para auditoría.
created_by	int(11)	Si	No	Campo para auditoría.
created_date	Timestamp	Si	No	Campo para auditoría.
last_modified_by	int(11)	Si	No	Campo para auditoría.
last_modified_date	Timestamp	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 14. Tabla persona

Descripción: contiene la información de la persona				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla persona.
id_tipo_identificacion	int(11)	No	No	FK de la tabla tipo_identificacion.
id_sexo	int(11)	No	No	FK de la tabla catalogo.
numero_identificacion	varchar(20)	No	No	Almacena el número de identificación.
nombre	varchar(250)	Si	No	Almacena el nombre.
apellido	varchar(250)	Si	No	Almacena el apellido.
fecha_nacimiento	Date	Si	No	Almacena la fecha nacimiento.
dirección	varchar(250)	Si	No	Almacena la dirección.
email	varchar(250)	Si	No	Almacena el correo electrónico.
subscripcion_mail	bit(1)	Si	No	Almacena la suscripción del mail.
activo	bit(1)	Si	No	Determina si el registro está activo.
versión	int(11)	Si	No	Campo para auditoría.
created_by	int(11)	Si	No	Campo para auditoría.

created_date	Timestamp	Si	No	Campo para auditoría.
last_modified_by	int(11)	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 15. Tabla proceso_vehiculo

Descripción: contiene la información de los procesos de los vehículos				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla proceso_vehiculo.
id_vehiculo	int(11)	No	No	FK de la tabla vehiculo.
id_tipo_proceso	int(11)	No	No	FK de la tabla tipo_proceso.
id_estado_proceso	int(11)	No	No	FK de la tabla estado_proceso.
descripcion	mediumtext	Si	No	Almacena la descripción.
id_usuario_ingresa	int(11)	Si	No	FK de la tabla usuario.
fecha_ingreso	Datetime	Si	No	Almacena la fecha de ingreso.
fecha_salida	Datetime	Si	No	Almacena la fecha de salida.
utiliza_autorizacion_retiro	bit(1)	Si	No	Almacena la autorización del retiro.
activo	bit(1)	Si	No	Determina si el registro está activo.
versión	int(11)	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 16. Tabla programacion_notificacion

Descripción: contiene la relación entre la notificación y tareas programadas				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla programacion_notificacion.
id_tarea_programada	int(11)	No	No	FK de la tabla tarea_programada.
id_notificacion	int(11)	No	No	FK de la tabla notificación.
id_grupo	int(11)	Si	No	FK de la tabla grupo.
id_usuario	int(11)	Si	No	FK de la tabla usuario.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 17. Tabla promocion

Descripción: contiene la información de las promociones				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla promoción.
id_tipo_promocion	int(11)	No	No	FK de la tabla catalogo.
descripción	varchar(70)	Si	No	Almacena la descripción.

cuerpo	Mediumtext	Si	No	Almacena el cuerpo.
inicio_vigencia	Timestamp	Si	No	Almacena el inicio de vigencia.
fin_vigencia	Timestamp	Si	No	Almacena el fin de la vigencia.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 18. Tabla respuesta_notificacion

Descripción: contiene las respuestas de las notificaciones				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla respuesta_notificacion.
id_programacion_notificacion	int(11)	No	No	FK de la tabla programacion_notificacion.
respuesta_boolean	bit(1)	Si	No	Almacena la respuesta boleana.
respuesta_texto	varchar(100)	Si	No	Almacena la respuesta en texto.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 19. Tabla tarea_programada

Descripción: contiene la tarea programada				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla tarea_programada.
id_tipo_tarea	int(11)	No	No	FK de la tabla tipo_tarea.
id_tipo_periodo	int(11)	Si	No	FK de la tabla tipo_periodo.
fecha_especifica	Date	Si	No	Almacena la fecha específica.
fecha_inicio	Date	Si	No	Almacena la fecha de inicio.
fecha_fin	Date	Si	No	Almacena la fecha de fin.
dia_numero	smallint(6)	Si	No	Almacena el día.
activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 20. Tabla tipo_trabajo

Descripción: contiene la información de tipo de trabajo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla tipo_trabajo.
id_categoria_ATAE	int(11)	No	No	Almacena la categoría ATAE.
id_unidad	int(11)	Si	No	FK tabla catálogo, almacena el tipo de unidad.
descripción	varchar(250)	Si	No	Almacena la descripción.

precio	decimal(10,2)	Si	No	Almacena el precio.
codigo_ATAE	varchar(10)	Si	No	Almacena el código ATAE.
tiempo_minimo	int(11)	Si	No	Almacena el tiempo mínimo de duración.
tiempo_maximo	int(11)	Si	No	Almacena el tiempo máximo de duración.
tiempo_aproximado	int(11)	Si	No	Almacena el tiempo aproximado de duración.
created_by	int(11)	Si	No	Campo para auditoría.
created_date	Timestamp	Si	No	Campo para auditoría.
last_modified_by	int(11)	Si	No	Campo para auditoría.
last_modified_date	Timestamp	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 21. Tabla usuario

Descripción: contiene la información de usuario				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria del usuario.
id_persona	int(11)	No	No	FK tabla persona.
id_categoria	int(11)	No	No	FK tabla catálogo, almacena el tipo de usuario.
id_codigo_push	varchar(250)	Si	No	Almacena el código de GCM.
apodo	varchar(20)	No	No	Almacena el nombre de usuario.
contrasenia	varchar(20)	Si	No	Almacena la contraseña.
numero_intento	int(11)	Si	No	Almacena el número de intentos permitidos al autenticarse.
activo	bit(1)	Si	No	Determina si el registro está activo.
versión	int(11)	Si	No	Almacena la versión del archivo.
created_by	int(11)	Si	No	Campo para auditoría.
created_date	Timestamp	Si	No	Campo para auditoría.
last_modified_by	int(11)	Si	No	Campo para auditoría.
last_modified_date	Timestamp	Si	No	Campo para auditoría.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 22. Tabla usuario_grupo

Descripción: contiene la relación entre usuario y grupo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
id	int(11)	No	Si	Clave primaria de la tabla usuario_grupo.
id_usuario	int(11)	No	No	FK tabla usuario.
id_grupo	int(11)	No	No	FK tabla grupo.
activo	bit(1)	No	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 23. Tabla usuario_vehiculo

Descripción: contiene la relación de usuario con vehículo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
Id	int(11)	No	Si	Clave primaria de la tabla perfil_usuario.
id_vehiculo	int(11)	No	No	FK de la tabla vehiculo.
id_usuario	int(11)	No	No	FK de la tabla usuario.
Activo	bit(1)	Si	No	Determina si el registro está activo.

Elaborado por: Paul Jara & Javier Rivera.

Tabla 24. Tabla vehiculo

Descripción: contiene la información del vehículo				
Campo	**Tipo**	**Nulo**	**PK**	**Descripción**
Id	int(11)	No	Si	Clave primaria de la tabla item_orden_trabajo
id_marca	int(11)	No	No	FK de la tabla marca.
id_tipo_vehiculo	int(11)	No	No	FK de la tabla catalogo.
modelo	varchar(100)	Si	No	Almacena el modelo.
kilometraje_mensual	int(11)	Si	No	Almacena el kilometraje mensual.
placa	varchar(10)	No	No	Almacena la placa.
anio	varchar(4)	Si	No	Almacena el año.
serie_motor	varchar(30)	Si	No	Almacena la serie del motor.
color	varchar(45)	Si	No	Almacena el color del vehículo.
chasis	varchar(45)	Si	No	Almacena el número de chasis.
activo	bit(1)	Si	No	Determina si el registro está activo.
versión	int(11)	Si	No	Campo para auditoria.

Elaborado por: Paul Jara & Javier Rivera.